Karlsruhe Institute of Technology

Advanced Calibration of Automotive Augmented Reality Head-Up Displays

To obtain an academic degree of
Doctor of Engineering (Dr.-Ing.)
from the
Department of Electrical Engineering and Information Technology
of the
Karlsruhe Institute of Technology (KIT)
accepted

DOCTORAL THESIS

of

M.Sc. Gao, Xiang

Examination date:	April 15, 2021
Supervisor:	Prof. Dr. rer. nat. Wilhelm Stork
Second supervisor:	Prof. Dr. rer. nat. Uli Lemmer

August 12, 2021

Karlsruhe Institute of Technology

Erweiterte Kalibrierung von Automotiven Augmented Reality-Head-Up-Displays

Zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
von der
KIT-Fakultät für Elektrotechnik und Informationstechnik
des
Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

M.Sc. Gao, Xiang

Tag der mündlichen Prüfung:	15. April 2021
Hauptreferent:	Prof. Dr. rer. nat. Wilhelm Stork
Korreferent:	Prof. Dr. rer. nat. Uli Lemmer

12. August 2021

Abstract

This thesis proposes advanced calibration methods for automotive augmented reality head-up displays (AR-HUDs) based on parametric perspective projection and nonparametric distortion models. The AR-HUD calibration is essential for rendering correctly placed and rectified virtual objects in the latest relevant applications, such as navigation and parking. Though state of the art has demonstrated some useful approaches, we are motivated to develop more advanced yet uncomplicated ones. As a prerequisite for the calibration, we defined several relevant coordinate systems, including the three-dimensional (3D) world, viewpoint space, HUD field of view (HUD-FOV) space, and the two-dimensional (2D) virtual image space. We describe the projection of images from an AR-HUD projector to the driver's eyes as a view-dependent pinhole camera model that consists of intrinsic and extrinsic matrices. Under this assumption, we first estimate the intrinsic matrix utilizing the boundary of HUD-FOV. Next, we calibrate the extrinsic matrices at different viewpoints inside a selected "eye box", considering drivers' changing eye positions. Those viewpoints' 3D positions are tracked using a driver camera. For any single view, we obtain a group of 2D–3D correspondences between a point array in the virtual image space and their matching control points in front of the windshield. Once these correspondences are available, we compute the extrinsic matrix at the viewpoint. By comparing the reprojected and real pixel positions of those virtual points, we acquire a 2D distribution of bias vectors, with which we reconstruct warping maps that contain the distortion information. For completeness, we repeat the above extrinsic calibration procedures at all the opted viewpoints. Using the calibrated extrinsic parameters, we restore the viewpoints in the world coordinate system. Since we have tracked them in the driver camera space simultaneously, we further calibrate the transformation from the driver camera to the world space by utilizing these 3D–3D correspondences. To deal with non-participating viewpoints inside the eye box, we obtain their extrinsic parameters and warping maps via nonparametric interpolation. Our combination of parametric and nonparametric models outperform state of the art concerning target complexities and time-efficiency, while maintaining a comparable calibration accuracy. Under all of our calibration schemes, the projection errors in the evaluation phase at 7.5 m distance fall within a few millimeters, which means an angular accuracy of about 2 arcminutes, which is close to the resolution of the eye.

Zusammenfassung

In dieser Arbeit werden fortschrittliche Kalibrierungsmethoden für Augmented-Reality-Head-up-Displays (AR-HUDs) in Kraftfahrzeugen vorgestellt, die auf parametrischen perspektivischen Projektionen und nichtparametrischen Verzerrungsmodellen basieren. Die AR-HUD-Kalibrierung ist wichtig, um virtuelle Objekte in relevanten Anwendungen wie z.B. Navigationssystemen oder Parkvorgängen korrekt zu platzieren. Obwohl es im Stand der Technik einige nützliche Ansätze für dieses Problem gibt, verfolgt diese Dissertation das Ziel, fortschrittlichere und dennoch weniger komplizierte Ansätze zu entwickeln. Als Voraussetzung für die Kalibrierung haben wir mehrere relevante Koordinatensysteme definiert, darunter die dreidimensionale (3D) Welt, den Ansichtspunkt-Raum, den HUD-Sichtfeld-Raum (HUD-FOV) und den zweidimensionalen (2D) virtuellen Bildraum. Wir beschreiben die Projektion der Bilder von einem AR-HUD-Projektor in Richtung der Augen des Fahrers als ein ansichtsabhängiges Lochkameramodell, das aus intrinsischen und extrinsischen Matrizen besteht. Unter dieser Annahme schätzen wir zunächst die intrinsische Matrix unter Verwendung der Grenzen des HUD-Sichtbereichs. Als nächstes kalibrieren wir die extrinsischen Matrizen an verschiedenen Blickpunkten innerhalb einer ausgewählten "Eyebox" unter Berücksichtigung der sich ändernden Augenpositionen des Fahrers. Die 3D-Positionen dieser Blickpunkte werden von einer Fahrerkamera verfolgt. Für jeden einzelnen Blickpunkt erhalten wir eine Gruppe von 2D–3D-Korrespondenzen zwischen einer Menge Punkten im virtuellen Bildraum und ihren übereinstimmenden Kontrollpunkten vor der Windschutzscheibe. Sobald diese Korrespondenzen verfügbar sind, berechnen wir die extrinsische Matrix am entsprechenden Betrachtungspunkt. Durch Vergleichen der neu projizierten und realen Pixelpositionen dieser virtuellen Punkte erhalten wir eine 2D-Verteilung von Bias-Vektoren, mit denen wir Warping-Karten rekonstruieren, welche die Informationen über die Bildverzerrung enthalten. Für die Vollständigkeit wiederholen wir die obigen extrinsischen Kalibrierungsverfahren an allen ausgewählten Betrachtungspunkten. Mit den kalibrierten extrinsischen Parametern stellen wir die Betrachtungspunkte wieder her im Weltkoordinatensystem. Da wir diese Punkte gleichzeitig im Raum der Fahrerkamera verfolgen, kalibrieren wir weiter die Transformation von der Fahrerkamera in den Weltraum unter Verwendung dieser 3D–3D-Korrespondenzen. Um mit nicht teilnehmenden Betrachtungspunkten innerhalb der Eyebox umzugehen, erhalten wir ihre extrinsischen Parameter und Warping-Karten durch nichtparametrische Interpolationen. Unsere Kombination aus parametrischen und nichtparametrischen Modellen übertrifft den Stand der Technik hinsichtlich der Zielkomplexität sowie Zeiteffizienz, während wir eine vergleichbare Kalibrierungsgenauigkeit beibehalten. Bei allen unseren

Kalibrierungsschemen liegen die Projektionsfehler in der Auswertungsphase bei einer Entfernung von 7,5 Metern innerhalb weniger Millimeter, was einer Winkelgenauigkeit von ca. 2 Bogenminuten entspricht, was nahe am Auflösungvermögen des Auges liegt.

Contents

List of Figures

List of Tables

1. Introduction

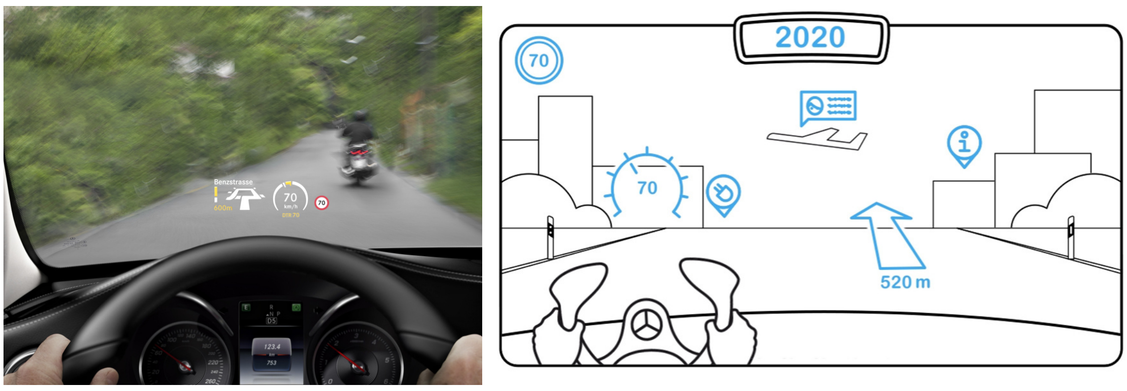
This chapter gives a brief introduction to the augmented reality head-up display (AR-HUD) from both the technology and application aspects, especially those related to the automotive industry. Section 1.1 provides a general background. Section 1.2 states the necessity in the calibration of AR-HUDs and relevant issues. Next, in Section 1.3, we review state of the art on this topic and point out their novelties and disadvantages. Section 1.4 profiles our contributions in this field. Finally, Section 1.5 provides readers an outline of the whole thesis.

1.1 Background

The head-up display (HUD) is an advanced human-machine interface (HMI) [1], which finds its applications in many modern transportation systems. It origins from aircraft [2]–[4] and renders much useful information to the pilot, e.g. flight-path markers and airspeed trend vectors. Images showing various real-time information are projected from the HUD projector and reflected by an optical combiner (e.g. the windscreen or a partially reflective device) installed in the cockpit until they reach the pilot's sight. They are defined as "virtual images" because they are reflected ones but not directly from an optical source existing in the external environment. With such a "see-through" display, the pilot can receive all the projected information without moving the sight to the dashboard. The appearance of HUD as an advanced avionic device has already enhanced the safety in taking off and landing operations of the aircraft [5], [6].

The HUD has become a hot topic in the automotive world since people proposed its potential application in vehicles [7], [8]. The HUD projector is embedded behind the main dashboard, and the windshield serves as the optical combiner. According to the presence of augmented reality (AR) technology, current on-vehicle HUDs are roughly categorized into two groups, i.e. conventional non-AR-HUDs, and more advanced AR-HUDs, as are illustrated in Figure 1.1. The former renders only static information that is not or indirectly related to the real-time environment, such as the driving speed, non-immersive navigation signs, or fuel consumption. Theoretically, these static symbols can appear at any region on the virtual image with proper color, brightness and contrast. Most HUDs available on the automotive market belong to this sort, which can be recognized as a "second" dashboard yet with enough transparency. The latter, i.e. AR-HUDs, generate dynamic virtual images that are highly immersive in the real world, such as immersive navigation signs, dynamic semantic labels, floating street names and house numbers. They are realized based on alignment between the projected virtual objects and the surrounding real objects.

There are critical benefits via using an on-vehicle AR-HUD. Firstly, as a HUD, it delivers real-time information via virtual images in the driver's sight, which is superior to the conventional dashboard in keeping the driver's attention on the traffic and reducing human eyes' accommodation time. Secondly, underlying vehicle sensor data, AR-HUDs can show important information graphically or semantically fused with the real objects, which provides a higher level of driving experience. In the driver's view, virtual images from AR-HUD look like natural extensions of the real ones. Furthermore, an AR-HUD can support more human-machine interaction when its immersive rendering is combined with advanced driver assistance systems (ADAS) and autonomous vehicle (AV) technology, like in lane recognition, advanced parking guidance and intelligent speed adaption. When the ADAS detect threats on the road, the alarming information can be directly placed in the driver's sight, followed by immediate decision and action. Last but not least, using AR cues can also improve awareness of hazardous traffic objects in low visibility, whereas that of non-hazardous ones is not affected, which is especially beneficial for elderly drivers [9]. In a nutshell, AR-HUDs make driving safer, more intelligent, more user-friendly and more enjoyable.



(a) An non-AR-HUD. (b) An AR-HUD.

Figure 1.1: Illustration of virtual images from a conventional on-vehicle non-AR-HUD[1] (a) and an advanced AR-HUD[2] (b). The non-AR-HUD renders a set of static information, including the current speed, the speed limit and a non-immersive navigation sign. The AR-HUD can also provide immersive navigation (the arrow looking like floating on the street) and dynamic virtual symbols for real objects (the building and the airplane).

1.2 Problem Statement

The calibration demand for AR-HUD comes from several aspects. First, since there are multiple sensors inside the vehicle, e.g. the 3D sensor, virtual camera (the driver's eyes) and driver camera (head tracker), the spatial relations between different coordinate systems should be accurately determined; otherwise, the virtual images can misalign the real world due to wrongly defined projection matrices or linear transformations, as is shown in Figure 1.2. Although these relations' target values are acquirable from vehicle design or production sections, the actual ones can be sensitive to the assembly tolerances of relevant components as the AR-HUD is an optical system. Hence, after all the components are installed on the vehicle assembling line, a calibration routine for AR-HUDs is still necessary. Furthermore, once any related device like the windshield, HUD projector, or head

[1]`https://media.daimler.com:443/marsMediaSite/Media/VXJO2UIhu7SfsONnt65cE61E4X4UUoYI46H8fgB2k4StjLRlBcdmtEA75Lr1Bj9t/48863007`, Retrieved 2021-02-01.

[2]`https://media.daimler.com:443/marsMediaSite/Media/c34s562D0v66gcP30b39HN500ADoYGqFHzR4YyH46ke4VA9wu6t76t0Zz8993au9/48862971`, Retrieved 2021-02-02.

tracker is repaired, replaced or updated in the aftermarket, the factory default settings may become no longer valid. Thus, the efforts in calibration are inevitable in production as well as aftermarket.

Second, since the windshield is not designed primarily for the reflection in HUD, but mainly for aerodynamics, aesthetics, or safety, its curved form leads to view-dependent optical distortions in projected virtual images [10], [11]. Though in some HUDs, their optics, e.g. reflective mirrors inside, are designed w.r.t. the windshield, residual distortion still exists and will affect the image quality. Meanwhile, each windshield or individual HUD projector may be subject to production-related deviations from the designed geometry. Consequently, distortion compensation also becomes an inevitable issue in the AR-HUD calibration.

Last but not least, the AR rendering is highly coupled with the real-time viewpoint, which is usually monitored by an inherently equipped driver camera. Both the projection of HUD images and optical distortion should change with the driver's eye positions. Therefore, the AR-HUD calibration should consider the acquisition of corresponding parameters at various viewpoints, without much sacrifice on either the precision or time- and cost-efficiency. A calibration method is valid and applicable only when it works well for any reasonable view pose.

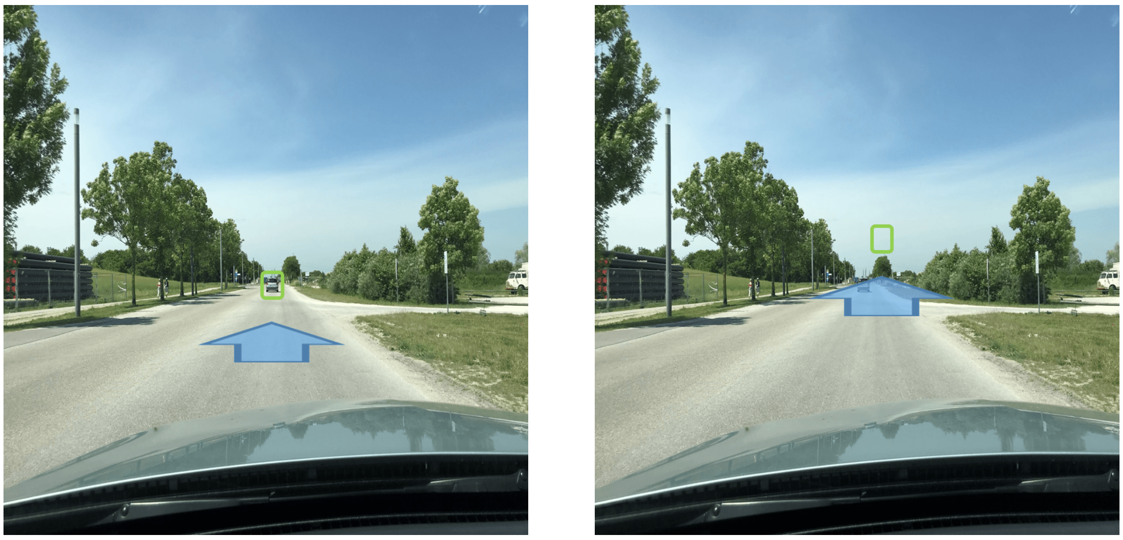

(a) Correct AR rendering. (b) Biased AR rendering.

Figure 1.2: Illustration of virtual objects (e.g. a blue navigation arrow and a green vehicle tracking box) presented by a precisely calibrated AR-HUD (a) and a biased one (b). Compared to the correct AR rendering via a calibrated HUD, that from an uncalibrated or wrongly calibrated HUD may lead to deviated navigation or occlusion.

Therefore, we have the following objectives:

- to realize robust calibration concepts that can be applied in both production line and after-sale scenarios in the automobile industry;
- to simplify the calibration methods, so that human labor can be saved to the greatest

extent;

- to qualitatively and quantitatively analyze factors that have an impact on the calibration implementation and results;
- to develop new approaches that make it possible for drivers to finish the calibration by themselves, without visiting the factory or workshop;
- if possible, to integrate the methods into actual vehicles and analyze the performance.

1.3 State of the Art

In this section, we introduce the previous work that is related to our main topic. To make it clear, we briefly review three research areas, i.e. calibration methods for cameras, AR-HUDs and AR-HMDs, respectively. The calibrations of cameras and AR-HMDs share similarities with the AR-HUD calibration, though they are applied in different instances than AR-HUDs.

1.3.1 Calibration of Cameras

Since in later chapters, we will adopt the pinhole camera model to describe our AR-HUD system, we review some literature about traditional camera calibration. There is an extensive body of literature about camera calibration, whether in computer vision or other photogrammetric applications. For short, we introduce here two popular methods, i.e. those proposed by Tsai [12] and Zhang [13]. These two methods can be implemented with a chessboard pattern as the calibration target that we also use in the calibration and evaluation phases later in Chapter 4.

Tsai's calibration model assumes that the manufacturer provides the camera's parameters to reduce the initial guess of the estimation. It requires n features points ($n > 8$) per image and solves the calibration problem with a set of n linear equations based on the radial alignment constraint. A second-order radial distortion model was used, while no decentering distortion terms were considered. The two-step approach copes with either a single image or multiple images of a 3D or planar calibration grid, but grid point coordinates must be known.

Zhang's calibration method requires people to place a planar chessboard at different orientations (more than 2) in front of the camera. The developed algorithm uses the chessboard pattern's extracted corner points to compute a projective transformation between the image points of the n different images, up to a scale factor. Afterward, the camera interior and exterior parameters are recovered using a closed-form solution, while the radial distortion terms are determined as a linear least-squares solution. A final nonlinear minimization of the reprojection error, solved using a Levenberg-Marquardt method (see also Section 2.10), refines all the retrieved parameters.

1.3.2 Calibration of AR-HUDs

Wientapper et al. [10] proposed a camera-based AR-HUD calibration method, which has already considered the influence of moving viewpoints. They employed the structure-from-motion (SfM) technique for precise camera registrations, assuming a view-independent virtual image plane. They estimated the projective transformation from the world to the calibration camera and compensated for the optical distortion by introducing a fifth-degree polynomial model. Their method has been adopted in systematical AR-HUD designs [14]. However, this approach requires extra effort in preparation. It is stated that most of the windshield area should be covered by highly textured patterns, except the HUD's field of view (HUD-FOV) that is covered by black canvas; otherwise, the feature points may be detected incorrectly. Besides, additional markers should be placed near or onto the windshield following the computer-aided design (CAD) model, which is necessary for calculating a scaled transformation from the feature map to the vehicle coordinates. Both the covers and markers are not initially equipped in cars, and they should be attached with enough precision by the staff. Thus, this approach may not be well suited for quick calibrations either in the production or aftermarket due to time-consumption or high precision requirements in attaching additional components. Additionally, the assumption of a view-independent virtual image plane may not hold since the optical path changes indeed with the viewpoint.

Hosseini et al. [15] published a systematical design for the AR-HUD using stereo night vision with an integrated calibration pipeline. Their method first solves an image warping matrix by iteratively minimizing a cost function, which can neutralize the optical distortion induced by the non-flat windshield surface. Later, they calibrated an external stereo thermal camera and a monocular night vision camera separately using a chessboard pattern. The stereo camera detects 3D real-world objects, while the monocular one tracks the driver's eye positions. They applied this system in displaying alert signs in case of collision against pedestrians under dark lighting conditions. When rendering the virtual objects, the stereo camera detects the obstacle in front of the vehicle, and the eye-tracking camera returns the driver's current line of sight. By solving the real-time spatial intersection of this line on the windshield surface, the AR-HUD knows where to place the virtual signs. However, this method requires a pre-calibration between the AR-HUD projector and the windshield surface. Besides, both solving the warping matrix or calibrating the involved cameras rely on pre-designed patterned targets. Empirically, the image warping matrix may vary with the viewpoints, which was not yet considered in their procedures.

Ueno and Komuro [16] raised another AR-HUD calibration method based on multi-view. For each viewpoint, they created a transformation map from the calibration camera image to the virtual image. Then they employed linear regression in the virtual image coordinates to estimate relevant coefficients and repeated this step for different viewpoints to create a look-up table (LUT). In their approach, the calibration camera has to gaze at the center of the virtual image, bringing difficulty in mounting and tuning the camera.

Yoon and Kim [17] developed a method based on the Scale-Invariant Feature Transform (SIFT) algorithm and Hough transformation. Using the SIFT algorithm, they detected and matched feature points in the environment ahead of the windscreen from both photos taken by a front view camera and a driver view (calibration) camera. By calculating the homography in between, they map the two camera images. Next, they display the rectangular virtual image frame and use Hough transformation to extract its four corners in the picture from the calibration camera. Then they acquire the homography from this camera to the HUD. Using these two homographies, they can project any 3D point sensed by the frontal view camera to the HUD virtual image. However, their method still has much room for improvement. The HUD's intrinsic parameters, such as focal lengths, were not estimated or measured. Moreover, there is no quantitative evaluation like in [16], [18]. Finally, they only used four corner points in the second step, which may lead to biased results when the optical distortion is noticeable.

Recently, Deng et al. [18] realized an AR-HUD calibration method using mixed reality glasses. In their work, a Microsoft HoloLens served as the calibration camera. Similar to [16], they mapped all pixels on the virtual image with those on the camera image, respectively. Then they formed a relatively large training dataset by sampling several hundreds of viewpoints and applied a nonlinear regression model for coefficients estimation, which is similar to that proposed by Wientapper et al. [10]. The calibration accuracy is validated with real-scene augmentations, while the calibration of the HoloLens w.r.t. the vehicle coordinate system is a prerequisite. To this end, they exploited the Iterative Closest Point (ICP) algorithm relying on the spatial mapping capability of HoloLens. However, a HoloLens is a piece of specialized equipment that may not be available in workshops. Though they also assert the full calibration process can be finished in one minute, the model training procedure also takes time. This time-consumption holds for each vehicle since either the configurations of HUD and windshield vary with the car model, or the mounting tolerance affects the optical path.

1.3.3 Calibration of AR-HMDs

Since AR-HUDs share similarities with projection-based AR-HMDs, the calibration methods of AR-HMD are also shortly reviewed here. For AR-HMD calibration, various solutions, such as the single point active alignment method (SPAAM) [19] and the interaction-free method [20]–[22] are two typical ways. However, the features of an automotive AR-HUD still differ from most AR-HMDs. The viewing angles of an AR-HUD are generally designed much narrower [14], [18], [23], [24], since the driver's eyes are usually focusing on scenes at a far more considerable distance than in an indoor environment, which brings a higher requirement on the angular calibration accuracy of HUD. For example, an 0.1° angular error can lead to nearly 90 mm length deviation at a distance of 50 m, i.e. over 6 % of typical vehicle heights.[3]

Meanwhile, since the HMD is wearable on the head, people can ignore the influence of head movement in its calibration but should take each eye's pose into account. Though some literature has considered individual pupils' motion, it still adopts physical models based on the premise of a view-independent static virtual image [20]. Recent work on

[3] `https://www.automobiledimension.com/car-comparison.php`, Retrieved 2018-11-24.

AR-HMD calibration also adopted nonparametric approaches, which has shown better accuracy than pinhole-based ones [25], [26]. However, these methods have limitations if applied to AR-HUD calibration. For example, the 3D space reconstruction in these approaches requires a large enough target screen covering the calibration camera's FOV, and this screen should be placed at several different distances. In this way, since AR-HUD's virtual image is located meters away, the screen size should also be up to several meters. That will result in both high costs and difficulties in preparing setups in factories or workshops, making such methods ultimately unsuitable for automotive applications.

1.4 Contributions

This dissertation concentrate on the calibration of AR-HUD systems. Although there may exist another AR display inside a vehicle, i.e. the video-based AR auxiliary display, its calibration methods are incredibly similar to those for conventional cameras. The AR-HUD calibration has its particular demands, as is mentioned in Section 1.2.

As is listed in Section 1.2, our main goal is to develop advanced applicable calibration methods for on-vehicle AR-HUDs. They should outperform those approaches in the previous work, at least on certain aspects. Here are the main contributions of this thesis:

- we propose various low-complexity calibration schemes, which are highly applicable in the automotive industry, both in the production and after-sales cases. These schemes cover the calibration using a sizable chessboard, a sheet of patterned paper, or inherent features from the environment, which also reflects the evolution of our practical implementations;
- we develop interpolation concepts to deal with multiple viewpoints, covering both the linear projection parameters and the nonlinear distortion models;
- all the involved mathematical and physical quantities are solved or approximately estimated in our calibration schemes, which is uncompleted in most of the previous work;
- we also finish different kinds of simulations, which are beneficial to our calibration experiments and future improvement;
- our calibration results are evaluated both qualitatively and quantitatively, demonstrating a comparable or even better accuracy than state of the art.

1.5 Outline

The rest of this dissertation is organized as follows: Chapter 2 introduces the mathematical and physical theories that are applied throughout the thesis, especially the pinhole camera model, perspective projection and epipolar constraint; Chapter 3 describes our AR-HUD system, including the involved devices, e.g. the HUD projector, calibration camera and windshield; Chapter 4 details the calibration methods that we have developed, which also reflect an evolution of experimental schemes; Chapter 5 shows the results of our calibration routines, including those from simulations and practical implementation; Chapter 6 compares different calibration approaches and discusses some specific issues; lastly, Chapter 7 draws a conclusion of this thesis and proposes an outlook for future work.

2. Theory

In this chapter, we will introduce the physical models and mathematical theories that are employed in our calibration work. In Section 2.1, we define all the involving coordinate systems, based on which we describe the projection and transformation matrices. Next in Section 2.2, we introduce shortly the rigid body motion and linear transformation. Section 2.3 explains basic principles in geometric optics, covering the reflection and refraction, which occur in the image rendering via an AR-HUD. Section 2.4 details the pinhole camera model in detail, which is a classical model to describe the linear perspective projection. It is the core theory to model AR-HUD systems throughout this dissertation. In Section 2.5, we introduce the stereo vision applied in the 3D sensing, particularly related to the estimation of HUD's focal lengths. Section 2.6 and 2.7 interpret the epipolar constraint and the SIFT algorithm respectively, both of which we utilize in the reconstruction of feature points for the target-free calibration approach (see also Section 4.5.3). Section 2.9 provides readers with the ideas to cope with the nonlinear optical distortion. Finally, Section 2.10 introduces the Levenberg-Marquardt Algorithm that we use to optimize the calibrated parameters.

2.1 Coordinate Systems

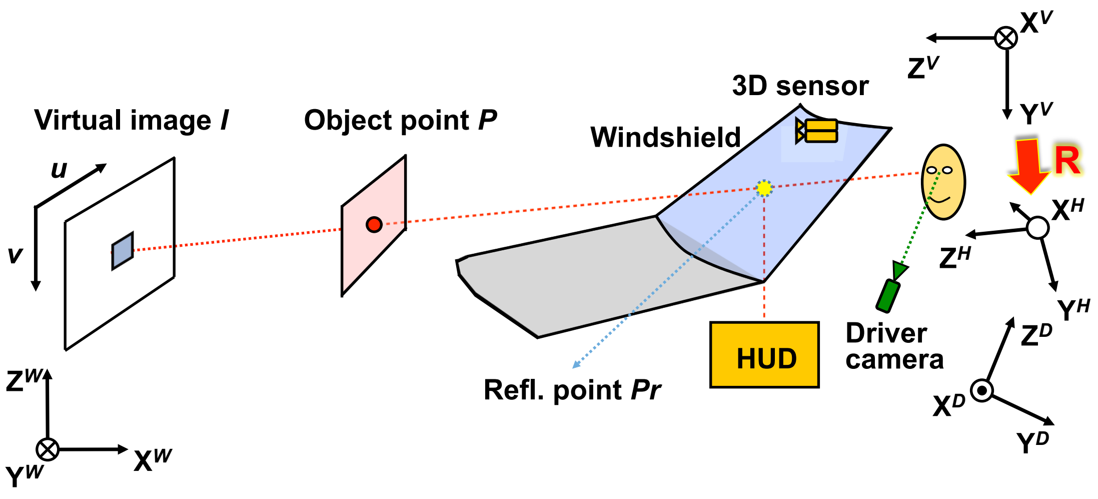


Figure 2.1: A side-view illustration of an automotive AR-HUD system. An optical ray departs from a HUD pixel and reaches the driver's sight via a reflection (refl.) point on the inner surface of the windshield, which corresponds to a virtual image pixel $[u, v]^T$. A driver camera serves as a real-time tracker for the viewpoint. The world space W, HUD-FOV space H, viewpoint space V and driver camera space D are marked with their $X-$, $Y-$ and $Z-$axes, respectively. $\mathbf{R}$ represents a relative rotation from the space V to H. Note that the optical path changes when the viewpoint moves.

Essentially an entire AR-HUD system involves the 3D world space W and the 2D virtual image plane I. As is sketched in Figure 2.1, an optical ray departs from a pixel in the HUD projector and reaches the driver's sight (indeed, a pupil of the two) via a reflection point on the inner surface of the windshield, which corresponds to a virtual image pixel. Thus, the driver receives the information presented on the virtual image. Since the optical path changes with the viewpoint, a tracking driver camera returns its real-time position.

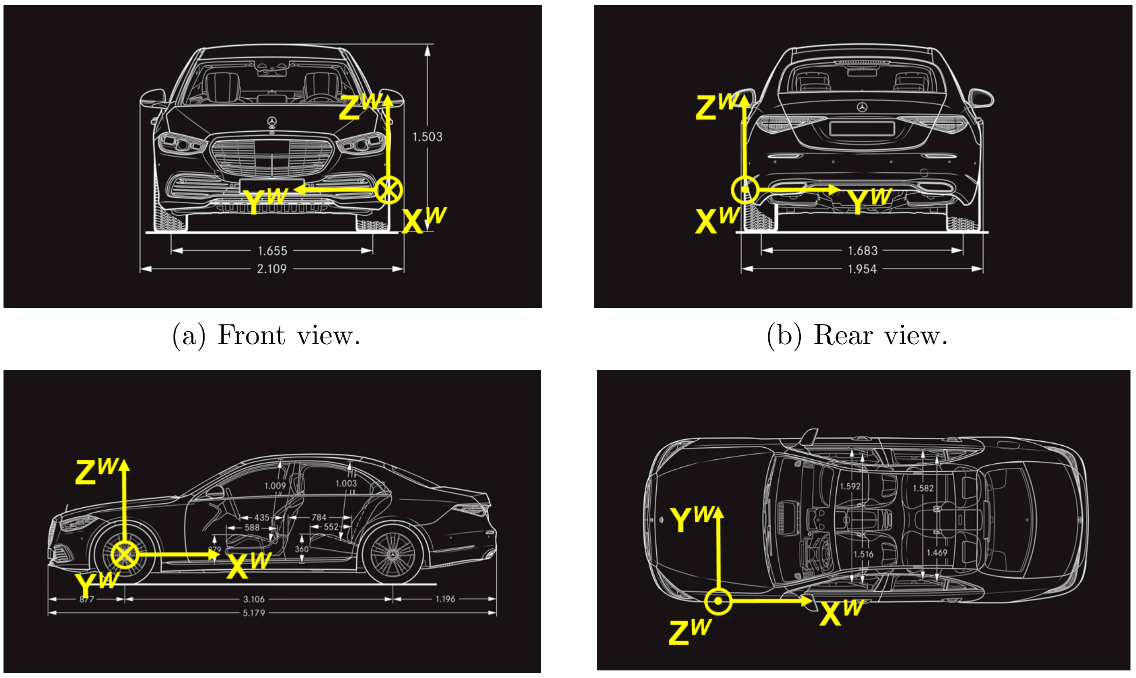
The primary goal of calibration for an AR-HUD system is to build up a general mapping relation between the 3D world squeezed in the HUD-FOV and the 2D virtual image. Other 3D positions outside the HUD-FOV are irrelevant because they cannot be projected onto the virtual image. If we consider the whole display area, such a corresponding relation is recognized as a 3D–2D surjection in mathematics. This means, for any valid 3D point $P_j = [X_j^W, Y_j^W, Z_j^W]^T$ in the world space W, there exists a unique 2D mapping point $p_{ij} = [u_{ij}, v_{ij}]^T$ on the virtual image I_i corresponding to the viewpoint V_i. On the contrary, any virtual point p_{ij} represents a single optical ray $\mathbf{r}_{ij}$ that originates from the pupil's optical center and passes through p_{ij}. Such an optical ray consists of infinitely many 3D points and extends to infinity. The surjection can be expressed as a spatial projection, a regression function, or even a simple look-up table, on which the rendering of virtual objects relies.

Hence, it is necessary to define the associated coordinate systems, i.e. spaces, no matter in which form this surjection is expressed. Except for the world space W and virtual image space I, there still involve other three 3D spaces, i.e. the HUD-FOV space H, the viewpoint space V, and the driver camera space D. We will detail them in the following sub-sections.

2.1.1 World Space

Taking the vehicle's movement into account, we define our world space W as a fixed space on the vehicle, instead of using an absolute geographic reference frame, e.g. that from a global navigation satellite system (GNSS). Therefore we also name it vehicle space. Theoretically, this coordinate system can be defined as an arbitrary Euclidean system. For example, we can locate the origin of coordinates at the centroid of a front wheel's outer disk for convenience. The X^W–, Y^W–, and Z^W–axes are illustrated in Figure 2.2:

In practice, real 3D position data from the frontal scene are returned via a 3D sensor fixed on the vehicle, such as a stereo camera, a light detection and ranging (LiDAR) apparatus, or an RGB-D camera. Here we assume that these sensors are pre-calibrated so that they already align the world space; otherwise, our calibration of AR-HUD loses the absolute accuracy. However, it still makes sense if there is only a systematical error in the 3D sensor, such as a rotational or translational bias. In this case, our AR-HUD is calibrated in a rotated or shifted world space W', on which the rendering of virtual objects is identically based. Alternatively, we can also directly employ the coordinate system of the equipped 3D sensor as the world space, which we, indeed, frequently adopt later in the laboratory environment with an off-vehicle AR-HUD setup. These sensors are becoming



(a) Front view.

(b) Rear view.

(c) Side view.

(d) Top view.

Figure 2.2: Schematic of our world space W in multiple views of a Mercedes-Benz S-Class Limousine WV223[1]vehicle. Here we locate the origin at the centroid of the left front wheel's outer disk.

more and more available in modern vehicles. Since our calibration usually happens in an offline condition, where the car is parked, we generally ignore the sensors' time delays.

2.1.2 Virtual Image Space

The virtual image plane I is governed by the projection characteristics of the HUD projector, as well as the optical combiner, i.e. the windshield in the automotive case. As is shown in Figure 2.1, it is buried in the frontal scene. The HUD projector delivers real images by emitting photons up to the windshield. Thus, the virtual image is the optical conjugate of the real one generated by the projector. The resolution of the virtual image depends on the arrangement of pixels inside the HUD projector.

As is sketched in Figure 2.1 and 2.3, we fix the origin of virtual image space at the upper left corner. Regardless of optical distortion, the u–axis stretches horizontally to the right and the v–axis vertically downwards in the driver's view. Nevertheless, due to the windshield's curved shape and reflective optics in the HUD projector, the virtual image plane I is also bent. This is an example of field curvature, which belongs to the optical distortions that we would like to compensate for during the calibration.

Since the receivers inside the HUD system are the driver's eyes, the optical imaging paths will change if they move. Under the effects of distortion, we anticipate that the virtual image plane I also changes slightly with the eye positions, which is later discussed

[1]Adapted using the pictures from `https://www.mercedes-benz.de/passengercars/mercedes-benz-cars/models/s-class/saloon-wv223/specifications/dimensions`, Retrieved 2021-02-02.

in Section 6.2. Therefore, the assumption of a static virtual image plane in previous work [10] is an approximation, though based on this point, their calibration method obtained an impressive accuracy.

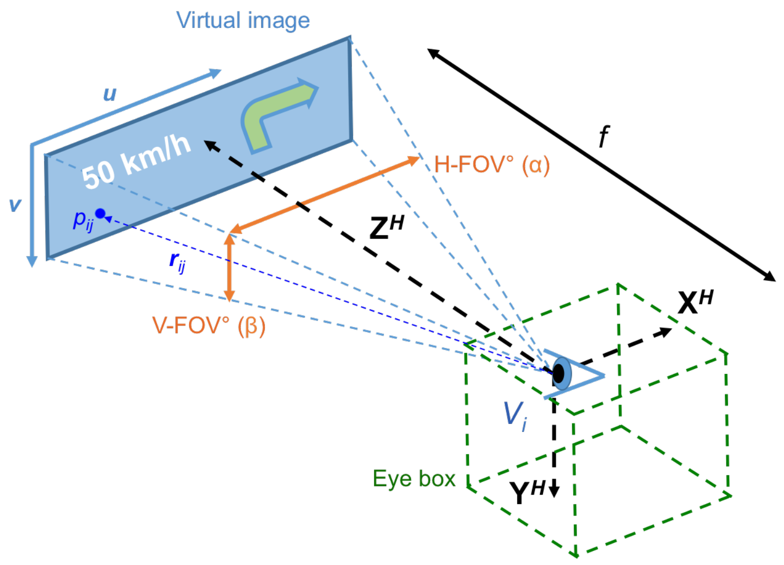


Figure 2.3: The FOV of a HUD, which is a 3D frustum. The origin is located at the viewpoint V_i, and the 2D virtual image space is marked by u– and v–axes. The horizontal (H-FOV°, i.e. α) and vertical (V-FOV°, i.e. β) opening angles determine its spans. The distance between the viewpoint and virtual image plane is the physical focal length f. The actual virtual image frame is slightly biased from a planar rectangular area due to optical distortion. p_{ij} represents a virtual point, while $\mathbf{r}_{ij}$ denotes its corresponding optical ray. The chief optical ray starts from the viewpoint and passes through the virtual image's central pixel. The X^H–, Y^H– and Z^H–axes label the 3D directions of HUD-FOV space. The eye box is a selected domain where the driver's viewpoint appears frequently.

2.1.3 HUD-FOV Space

The HUD-FOV, as is illustrated in Figure 2.3, is a frustum whose vertex lies at the viewpoint. In our implementation, the image receiver can be the driver's (or the operator's) eyes or a monocular calibration camera. It is located at a viewpoint inside a reasonable spatial domain named as "eye box", which is detailed later in Section 3.2. Conventionally, in the case of human eyes, the viewpoint is assumed to be at the middle point between the two pupils [10], [24], [27]. Under the assumption of an imperfect rectangular virtual image frame I, the HUD-FOV is a pyramid whose edges are the four rays starting from the viewpoint and passing through the virtual image's corners. Theoretically, these edges are at infinity. For convenience in description, we mark two spanning angles of this frustum, i.e. the horizontal (α) and vertical (β) opening angles.

Analogously, the HUD-FOV space H is defined as a 3D space based on the frustum as mentioned above. The origin is located at the vertex, and the X^H– and Y^H–axes go parallel to the u– and v–axes of the virtual image I. The Z^H–axis intersects the virtual image center. This HUD-FOV space serves as a transitional coordinate system in formulating the entire pinhole camera model in Section 2.4. Note that when the viewpoint is changed, the origin of this space shifts correspondingly.

2.1.4 Viewpoint Space

The viewpoint space V is, as is illustrated in Figure 2.1, a 3D space orthogonally transformed from the world W with right angles, i.e. $90°$ or $-90°$. It also serves as an intermediary in formulating the pinhole camera model. Later, we will assume that under the pinhole model, the viewpoint space V shares the same origin with the HUD-FOV space H. However, due to the HUD and windshield design, there may exist a relative rotation between them. In the viewpoint space, the X^V–axis points to the right, the Y^V–axis points to the floor, and then Z^V–axis points forward, which follows the typical definition of a camera coordinate system [13], [28], [29]. This is analogous to when we place a calibration camera at the viewpoint, which faces the frontal scene. Hence, the viewpoint space is beneficial when we use a calibration camera at the viewpoint instead of human eyes (see Section 4.4.2). Like the HUD-FOV space H, when the viewpoint moves, the origin of this space V varies accordingly.

2.1.5 Driver Camera Space

A driver camera is also included in the system, as is illustrated in Figure 2.1. We also name it head tracker because it can track the driver's head by detecting facial features. Like the viewpoint space V, the definition of this space follows the conventional camera coordinate system. However, sometimes an on-vehicle driver camera is already pre-calibrated to the world space W since its installation.

The driver camera's basic functionalities are demonstrated in both the calibration of AR-HUD and the afterward AR content rendering while driving. In the former process, this sensor measures the viewpoint position so that we can calibrate the corresponding view-dependent parameters. In the AR rendering, it provides the driver's current eye positions so that the picture generation unit (PGU) can call the corresponding parameters to project real objects on the correct pixel positions.

Later in Section 4.2, we will introduce our method to recalibrate this driver camera based on the calibration results of the AR-HUD, which is applicable in the after-sale case. Nevertheless, the driver camera's calibration is not our primary concern because it plays an implicit role in the AR-HUD calibration, though this camera is a necessary component in a vehicle with an AR-HUD system.

2.2 Rigid Body & Transformation

We assume that the whole vehicle is regarded as a rigid body, including the HUD projector, the connected sensors, and the windshield. This assumption means that we ignore the components' deformation, e.g. the small shape change when the car is vibrating on the road. It is reasonable in practical terms: on the one hand, the deformation should be small because of the rigidity of the material for vehicle production; on the other hand, it is impractical to model these deformations under any static calibration routine. The assumption of rigid bodies guarantees that the offline (when the vehicle is parked) modeling based on linear transformation can also be effective for online (when the vehicle runs on the road) AR-rendering. Of course, it is undeniable that in the future, there might be a new online calibration routine for the AR-HUD, i.e. a real-time implementation while the car is moving on the street. Nevertheless, it already makes much sense if we primarily accomplish robust offline calibration in the factory or workshop environment.

As a convention for the projective geometry, we adopt the homogeneous coordinates [30] throughout this thesis. That means we expand N–dimensional Cartesian coordinates with an additional element “1”, e.g. :

$$\begin{cases} \mathbf{v} = [v_1, v_2, v_3]^T \longrightarrow \mathbf{v} = [v_1, v_2, v_3, 1]^T \,, \\ \mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \longrightarrow \mathbf{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \end{cases} \tag{2.1}$$

where $\mathbf{v}$ (or $\mathbf{M}$) was originally an arbitrary vector (or 3×3 matrix) under Cartesian coordinates. The homogeneous coordinates bring convenience later in describing the perspective projection and formulating the pinhole camera model. Based on the rigid body assumption, we use Eq. (2.2) to express arbitrary 3D linear transformations, including rotation and translation.

$$\begin{cases} \mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{T}(\mathbf{t}) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \end{cases} \tag{2.2}$$

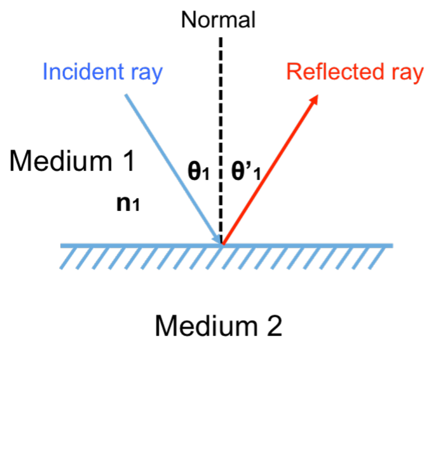
Here α, β and γ are arbitrary rotation angles, while $\mathbf{t} = [t_x, t_y, t_z]^T$ is an arbitrary 3D translational vector. These angles are also named the roll, pitch and yaw angles, respectively [31]. Hence, our 3D transformation $\mathbf{T}(\alpha, \beta, \gamma, \mathbf{t})$ uses the following paradigm:

$$\mathbf{T}(\alpha, \beta, \gamma, \mathbf{t}) = \mathbf{T}(\mathbf{t})\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{T}(\mathbf{t})\mathbf{R}_z(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_x(\alpha) \tag{2.3}$$

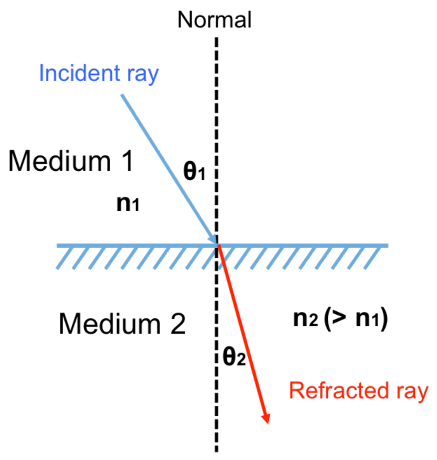
The meaning of Eq. 2.3 is: suppose we have a source coordinate system A and a destination coordinate system A', we first rotate the axes of A so that they go parallel to those of A', and then displace the origin. This paradigm is in line with the extrinsic matrix of the pinhole camera model in Section 2.4.

2.3 Ray Optics, Reflection & Refraction

Display and its calibration are generally related to geometric optics, or rather ray optics. In this context, we ignore the light's electromagnetic properties, such as spectrum or nonlinearity. In classical physics, the light shall go along an extreme as is described by Fermat's principle, which is usually the shortest way [32]. If we assume that the air is a homogeneous medium with a relative refractive index n_{air} close to 1, then the light shall propagate along with straight rays until it is reflected or enters another medium with a different refractive index. The assumption of ray optics is a generic prerequisite in many augmented reality applications.



(a) Reflection.



(b) Refraction.

Figure 2.4: Optical reflection (a) on the boundary between two media, and refraction (b) from an optically thinner to an optically denser medium.

Figure 2.4 sketches the cases of reflection and refraction. The illustrations are on the sectional plane formed by the incident ray and the normal line perpendicular to the boundary between the media. In the reflection case, an incident ray encounters a reflective interface and exits in a bounced manner with the angle of reflection (θ_1') equal to the incident angle (θ_1). In the case of refraction, the incident light penetrates the boundary, but with an angle of refraction (θ_2) that is different from the incident angle. Snell's law determines this angle:

$$n_1\theta_1 = n_2\theta_2, \tag{2.4}$$

where n_1 and n_2 are the refractive indices of the Medium 1 and 2, respectively. The reflection and refraction phenomenons can be further interpreted by Huygens–Fresnel principle [32], assuming that the monochromatic light is a series of propagating electromagnetic waves. Note that the two behaviors co-occur if the two media are both optically transparent at the incident light's wavelength. The intensities and polarizations of reflected and refracted components can be analyzed using the Fresnel equations [32].

2.4 Pinhole Camera Model

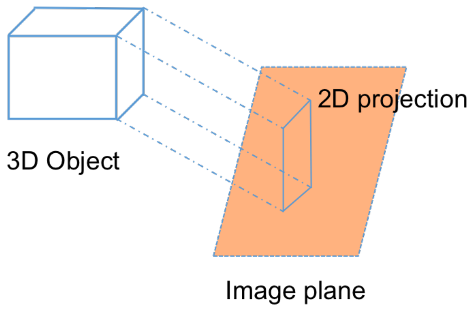
In the AR rendering for driving scenarios, we recognize the driver's viewpoint as the middle point between the two pupils. Indeed, since the human interpupillary distance is usually 50 mm to 75 mm [33], the human stereo vision can have an impact on the calibration. Ideally, for an individual driver, the AR-HUDs should be calibrated separately for both eyes. Nevertheless, this is impractical for the factory or workshop because people have various interpupillary distances and dominant eyes, and the workload and time consumption will double. Moreover, the physical focal length of automotive AR-HUD is usually designed up to over 10 m, which is significantly larger than the human interpupillary range. Objects to be augmented at this distance demonstrate tiny disparities in the human stereo vision. However, the above-stated effect shall be taken into account for other AR applications, such as indoor AR-HMD devices, since there are two separate displays for the two eyes. Alternatively, we can replace the human viewer with a real calibration camera, such as a smartphone camera or a monocular webcam.

2.4.1 Perspective Projection

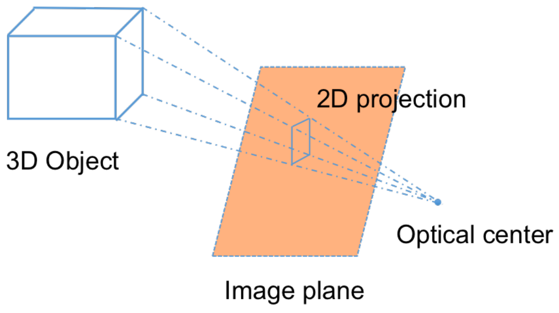
Perspective projection is a sort of linear projections where 3D objects are projected on a 2D picture plane. As the pinhole camera model's basis, it brings the effect that distant objects appear smaller than nearer objects even though they possess the same size. There is an optical center in the perspective projection, which is different from parallel projections, as is sketched in Figure 2.5. It also means that infinitely long parallel lines (e.g. roadsides in Figure 1.2 (a) and (b)) appear to intersect in the projected image at a vanishing point [34]. Photographic lenses [35] and the human eye [36] work in the same way. Therefore perspective projection looks most realistic. The pinhole camera model is a model to describe the simplest perspective projection scenario.

2.4.2 Formulation

Conventionally, a lens-based camera or imaging system can be modeled as a pinhole camera, as is illustrated in Figure 2.6. In the pinhole model, the center of the lens is recognized as a pinhole. Behind the pinhole is a black box, whose bottom at the focal length distance



(a) Parallel projection.



(b) Perspective projection.

Figure 2.5: Parallel (a) and perspective (b) projections. Compared with the former, there is an optical center in the latter, where all the projective optical rays meet each other.

is regarded as the image plane. The pinhole position is also the optical center, through which all the captured rays travel from the objects to the image plane. Under this model, it is convenient to formulate the imaging system using matrices and linear transformations. A typical application is the perspective projection in a monocular camera system.

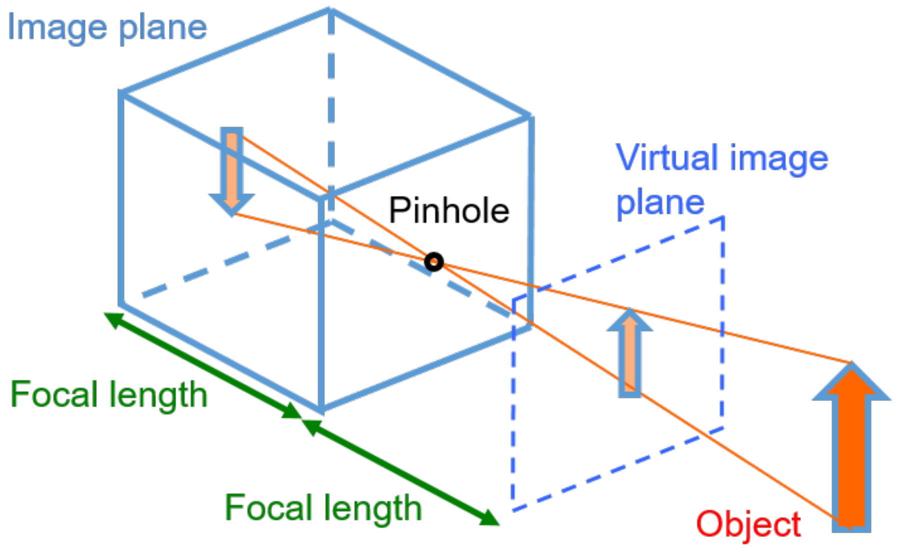


Figure 2.6: Pinhole camera model. A 3D object is projected to the real image plane at the focal length distance through a pinhole. A virtual image plane is located symmetrically to the real one.

Here we approximate the AR-HUD imaging system using this classical pinhole camera model. The pinhole model brings several advantages. First of all, it is relatively simple and includes only a few unknowns, which restrains the calibration approaches' complexity, especially compared to those using pure transformation mappings [16]. Secondly, since it is already widely studied in the context of camera calibration, various algorithms [29], [37]–[40] are available to solve the unknown parameters. Finally, applying the obtained calibration data in rendering augmentations is straightforward and can be readily developed upon existing software and hardware functionalities. Thus, we express the AR-HUD's pinhole model for a viewpoint indexed by i using homogeneous coordinates:

$$w \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} X^{H_i} \\ Y^{H_i} \\ Z^{H_i} \\ 1 \end{bmatrix} = \mathbf{P}_{I_i V_i} \begin{bmatrix} X^{V_i} \\ Y^{V_i} \\ Z^{V_i} \\ 1 \end{bmatrix} = \mathbf{P}_{I_i W} \begin{bmatrix} X^{W} \\ Y^{W} \\ Z^{W} \\ 1 \end{bmatrix}, \tag{2.5}$$

$$\mathbf{P}_{I_i V_i} = \mathbf{K}\mathbf{R}_{H_i V_i}, \tag{2.6}$$

$$\mathbf{P}_{I_i W} = \mathbf{K}\mathbf{T}_{H_i W} = \mathbf{K}\mathbf{R}_{H_i V_i}\mathbf{T}_{V_i W}, \tag{2.7}$$

$$\mathbf{K} = \begin{bmatrix} f_u & s & u_0 & 0 \\ 0 & f_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.8}$$

$$\mathbf{T}_{H_i W} = \mathbf{R}_{H_i V_i}\mathbf{T}_{V_i W} = \begin{bmatrix} \mathbf{R}_{H_i W} & \mathbf{t}_{H_i W} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.9}$$

In Eq. (2.5), $P = \left[X^W, Y^W, Z^W\right]^T$ represents an arbitrary point inside the world space W, which falls in the HUD-FOV, and $p_i = [u_i, v_i]^T$ is the corresponding pixel in the virtual image I_i. The number w is a scaling factor. The intrinsic matrix $\mathbf{K}$ in Eq. (2.8) describes the inherent characteristics of the HUD, such as respective focal lengths f_u and f_v (unit: pixel) in horizontal and vertical directions, principal point $[u_0, v_0]^T$ (unit: pixel) and the skew factor s. For a single point's projection, we have $w = Z^{H_i}$. These intrinsic parameters are determined by the HUD design and independent of the environment under non-extreme conditions. In contrast, the extrinsic matrix $\mathbf{T}_{H_i W}$ in Eq. (2.9), also named camera pose, defines a linear transformation that convert the point from the world W to the HUD-FOV space H_i. It consists of a 3×3 rotation matrix $\mathbf{R}_{H_i W}$ and a 3×1 translation vector $\mathbf{t}_{H_i W}$. Since the degree of freedom (DoF) of a rotation matrix is three, we can express it with the roll, pitch and yaw angles, as is defined in Section 2.2. Or we can convert it into a 3×1 Rodrigues rotation vector $\mathbf{r}_{H_i W} = [r_{1,H_i W}, r_{2,H_i W}, r_{3,H_i W}]^T$. The conversion is useful later in our interpolation concepts, with which we aim to spread our calibration results at selected viewpoints to other non-participating viewpoints (see also Section 4.1.4). Since the DoF of a 3D translation vector is also three, the extrinsic matrix has a DoF equal to six.

The 3×4 projection matrix $\mathbf{P}_{I_i V_i}$ in Eq. (2.5) and (2.6) projects a 3D point in the viewpoint space V_i to the virtual image I_i, which can be decomposed to the intrinsic matrix $\mathbf{K}$ and a 4×4 homogeneous rotation matrix $\mathbf{R}_{H_i V_i}$. Note that here we assume that under the pinhole model, the HUD-FOV space H_i and viewpoint space V_i share the same origin that is, indeed, the optical center. The other projection matrix $\mathbf{P}_{I_iW}$ in Eq. (2.5) and (2.7) projects a 3D point in the world space W to the virtual image I_i. As is implied above, it consists of the intrinsic matrix $\mathbf{K}$ and extrinsic matrix $\mathbf{T}_{H_iW}$. Note that $\mathbf{T}_{H_iW}$ can be further decomposed to the relative rotation $\mathbf{R}_{H_iV_i}$, and the transformation matrix $\mathbf{T}_{V_iW}$ that convert a point in the world space W into a viewpoint space V_i. These decompositions are so meaningful that later we will develop different calibration schemes based on them.

The involving spaces and the corresponding matrices are summarized in Figure 2.7. Note that the matrix $\mathbf{T}_{WD}$ represents the transformation from the driver camera space D to the world space W. Our parametric part of AR-HUD calibration is to restore all of these linear mathematical relations.

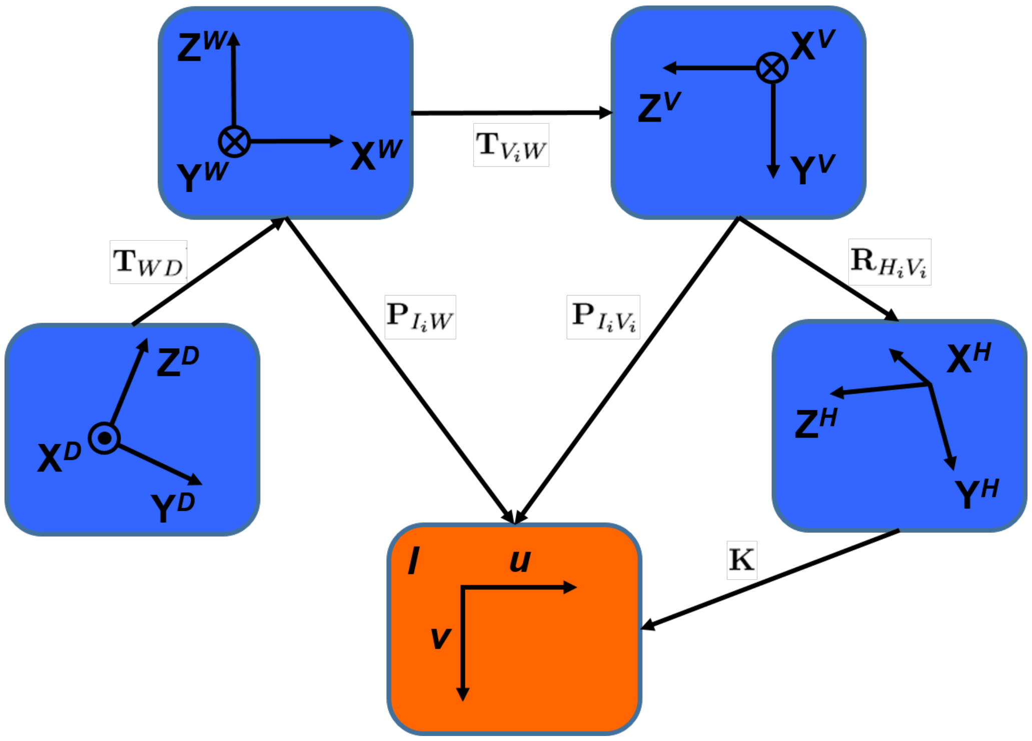


Figure 2.7: Linear relations among different involving 2D (orange) and 3D (blue) spaces, including the world space W, viewpoint space V, HUD-FOV space H, driver camera space D and virtual image space I. These spaces are pairwise connected by projection ($\mathbf{P}_{I_iW}$, $\mathbf{P}_{I_iV_i}$), transformation ($\mathbf{T}_{V_iW}$, $\mathbf{T}_{WD}$), rotation ($\mathbf{R}_{H_iV_i}$) or intrinsic ($\mathbf{K}$) matrices in the pinhole camera model. The subscript i indicates the view-dependency.

2.4.3 Simple Analysis

For an arbitrary 3D point $P = \left[X^W, Y^W, Z^W\right]^T$ within the HUD-FOV, we analyze here the impact of translational vector $\mathbf{t} = [t_1, t_2, t_3]^T$ according to the above pinhole model. For simplicity, we assume that:

- the HUD has identical horizontal and vertical focal lengths, i.e. $f_u = f_v = f$. This means the astigmatism is ignored;

- the skew factor s is negligible since the pixels of HUD projector usually have rectangular shape;
- the origin of image space I_i lies at the principal point, i.e. $[u_0, v_0]^T = [0, 0]^T$, which is different from the definition in Section 2.1.2.

Then the corresponding ground truth projected virtual point p_i is:

$$p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{fr_{11}X^W + fr_{12}Y^W + fr_{13}Z^W + ft_1}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + t_3} \\ \frac{fr_{21}X^W + fr_{22}Y^W + fr_{23}Z^W + ft_2}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + t_3} \\ 1 \end{bmatrix} . \tag{2.10}$$

If there is an error in the component t_1, i.e. t_1 becomes $t_1 + \Delta t_1$, then we have:

$$p_{i,\Delta t_1} = \begin{bmatrix} u_i + \Delta u_{i,\Delta t_1} \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{fr_{11}X^W + fr_{12}Y^W + fr_{13}Z^W + f(t_1 + \Delta t_1)}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + t_3} \\ \frac{fr_{21}X^W + fr_{22}Y^W + fr_{23}Z^W + ft_2}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + t_3} \\ 1 \end{bmatrix} , \tag{2.11}$$

which means that only u_i is influenced. That is, it leads to a horizontal shift of the virtual point p_i according to the sign of Δt_1. Similarly, an error in the component t_2 result in a vertical shift of p_i. However, when there is an error in t_3, we have:

$$\begin{aligned} p_{i,\Delta t_3} = \begin{bmatrix} u_i + \Delta u_{i,\Delta t_3} \\ v_i + \Delta v_{i,\Delta t_3} \\ 1 \end{bmatrix} &= \begin{bmatrix} \frac{fr_{11}X^W + fr_{12}Y^W + fr_{13}Z^W + ft_1}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + (t_3 + \Delta t_3)} \\ \frac{fr_{21}X^W + fr_{22}Y^W + fr_{23}Z^W + ft_2}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + (t_3 + \Delta t_3)} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} u_i \cdot \left(1 - \frac{\Delta t_3}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + (t_3 + \Delta t_3)}\right) \\ v_i \cdot \left(1 - \frac{\Delta t_3}{r_{31}X^W + r_{32}Y^W + r_{33}Z^W + (t_3 + \Delta t_3)}\right) \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} s_{i,\Delta t3} \cdot u_i \\ s_{i,\Delta t3} \cdot v_i \\ 1 \end{bmatrix} , \end{aligned} \tag{2.12}$$

where $s_{i,\Delta t3}$ is a real scaling factor subject to $s_{i,\Delta t3} \neq 1$. It means that the entire virtual image I_i is zoomed in or out jointly in the u– and v–directions, corresponding to the sign of Δt_3.

Nevertheless, the above analysis cannot cover all the possible factors affecting AR-HUDs' imaging or calibration, e.g. 2D/3D detection errors or tolerance of viewpoint positions. Unlike the errors in translational components, the others are difficult to derive explicitly from Eq. (2.5)–(2.9). However, they will be investigated with numerical simulations later in Section 5.2.

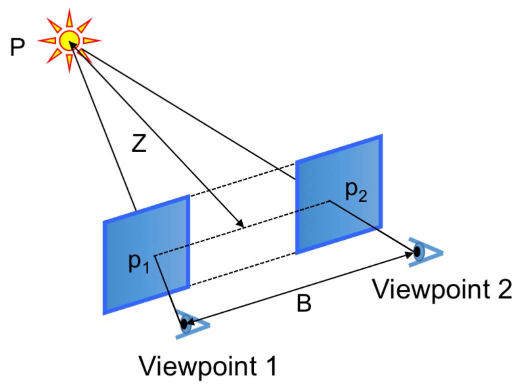
2.4.4 PnP Problem

The perspective-n-point (PnP) problem [41] is the matter of determining the camera pose (extrinsic matrix) in the pinhole camera model, with n presenting the number of applicable 2D–3D point correspondences between the image pixels and the world. Originally raised from camera calibration, the PnP problem is associated with many computer vision applications, including pose estimation and augmented reality [42], [43]. Since the camera pose consists of six DoF, we need at least six equations to solve it. According to Eq. (2.5), it is required $n \geq 3$ so that there are at least three equations accessible for both u– and v–directions. In such an extreme case, it becomes a P3P problem.

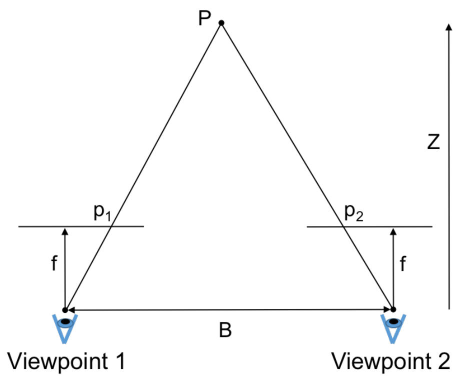
Some sophisticated calculation methods are already integrated into open-source software, such as the *OpenCV* Library [44]. Later we will intensively convert the calibration down to this PnP problem and try to solve it with adaptively modified algorithms.

2.5 Stereo Vision

The stereo vision theory is the basis for our 3D sensing because we will use a stereo camera as the external sensor, which measures the 3D positions in the forward scene. We will also construct a stereo vision scenario to estimate our HUD's intrinsic parameters in the automatic calibration schemes. Besides, when we use a stereo camera as the driver camera to track the viewpoint (see Section 3.6), it is again based on the principles in this section.



(a) Side view.



(b) Top view.

Figure 2.8: Stereo vision (a) including two parallel placed identical mono-cameras at Viewpoint 1 and 2 with a baseline distance B. P is a 3D point, and p_1 and p_2 are the corresponding projections in the two images. The depth Z can be triangulated (b) using the geometrical relations between similar triangles.

Let us assume a dual-camera-based stereo camera with two parallel positioned monocular "eyes", as is shown in Figure 2.8. Ideally, the two cameras have identical intrinsic parameters, e.g. focal length f in pixel, and principal point $[u_0, v_0]^T$ at the image center. They face along the same direction but are separated by a baseline distance B. Hence, the same 3D position P are imaged at different pixel positions $p_1 = [u_1, v_1]^T$ and $p_2 = [u_2, v_2]^T$

in the captured pictures. Because the cameras are placed at the same height, we have:

$$u_1 - u_2 = d, \quad v_1 = v_2, \tag{2.13}$$

where d is named disparity. According to the geometrical relations between similar triangles, we can further derive:

$$Z = \frac{Bf}{d}, \tag{2.14}$$

where Z is the depth from the baseline to the 3D point P. Let us further define a camera coordinate system whose origin is fixed at the left-eye camera's optical center. Then, the transversal components X and Y can also be determined if we apply the definition of axes in the viewpoint space V (Figure 2.1) here:

$$X = \frac{u - u_0}{f} Z, \quad Y = \frac{v - v_0}{f} Z. \tag{2.15}$$

Underlying these relations, a stereo camera, consisting of two such identical monocular cameras, can measure selected objects' 3D positions in the forward environment. Note that the depth detection error of a stereo camera is modeled as:

$$\partial Z = -\frac{Z^2}{Bf} \partial d. \tag{2.16}$$

This means a larger baseline distance helps reduce the error. In the popular KITTI dataset for mobile robotics and autonomous driving research [45], the authors used a stereo camera with $B \approx 54\,\mathrm{cm}$. However, this distance is often restricted by the available room inside or on top of the vehicle.

2.6 Epipolar Constraint

The epipolar constraint is derived from multiple view geometry and has many applications in computer vision [46]. Suppose we have two pre-calibrated monocular cameras, i.e. Camera 1 and 2, which provide us with two views facing the frontal scene, as is schetched in Figure 2.9. In machine vision, the essential matrix $\mathbf{E}$ can describe the relationship between the views:

$$\mathbf{E} = \hat{\mathbf{t}}\mathbf{R}, \quad \hat{\mathbf{t}}\mathbf{R} = \mathbf{t} \times \mathbf{R}, \tag{2.17}$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}, \quad \hat{\mathbf{t}} = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.18}$$

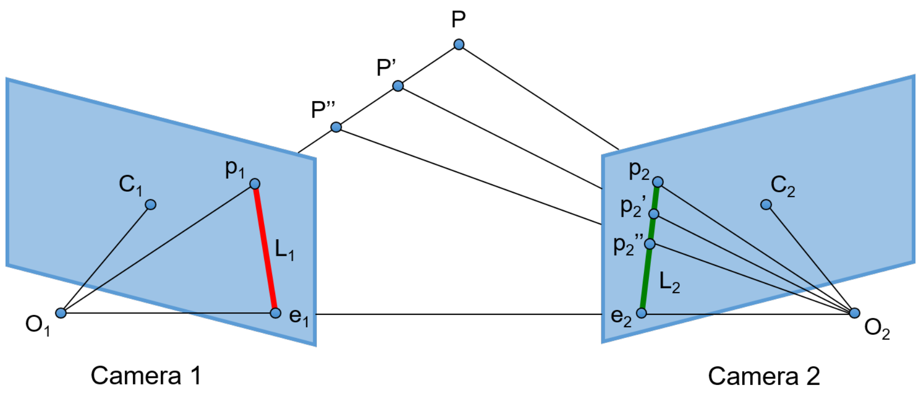


Figure 2.9: Epipolar constraint in a dual-camera case, where O_1 and O_2 are the cameras' optical centers, and C_1 and C_2 are the centers of images. P, P' and P'' are different 3D points on the same optical ray starting from O_1, generating p_1 as the projection in Camera 1, and p_2, p_2' and p_2'' as the projections in Camera 2. e_1 and e_2 are the intersections of the line O_1O_2 with the two image planes. L_1 and L_2 represent the epipolar lines.

where $\mathbf{t}$ is the translation vector and $\mathbf{R}$ is the rotation matrix, both of which compose the homogeneous transformation matrix $\mathbf{T}$ between these two camera spaces (from Camera 2 to 1). Furthermore, any homogeneous 2D–2D pixel correspondence p_1 and p_2 (suppose the two image spaces' origins are located at the image centers) in the two images follows the epipolar constraint:

$$p_2^T \mathbf{F} p_1 = \begin{bmatrix} u_2 & v_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = 0, \tag{2.19}$$

where $\mathbf{F}$ is called the fundamental matrix. The geometric interpretation of Eq. (2.19) is that p_2 must lie on the epipolar line defined by $\mathbf{F}p_1$ since p_1 and p_2 are the projections of the same 3D point P. By rearranging the equation, we convert the calculation of $\mathbf{F}$ to a least square problem:

$$\mathbf{x}^s \cdot \mathbf{f}^s = \begin{bmatrix} u_2u_1 & u_2v_1 & u_2 & v_2u_1 & v_2v_1 & v_2 & u_1 & v_1 & 1 \end{bmatrix} \cdot \mathbf{f}^s = 0, \tag{2.20}$$

where $\mathbf{f}^s$ is a degenerated vector form of $\mathbf{F}$:

$$\mathbf{f}^s = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}^T. \tag{2.21}$$

For n multiple point correspondences, the vectors $\mathbf{x}_i^s$ are stacked row-wisely into a $n \times 9$ matrix $\mathbf{X}^s$ and Eq. (2.20) becomes:

$$\mathbf{X}^s \cdot \mathbf{f}^s = \mathbf{0}, \tag{2.22}$$

which can be solved by least square algorithms. Then, we convert the fundamental matrix to an initial guess essential matrix with the pre-calibrated 3×3 intrinsic matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ of the two cameras:

$$\mathbf{E}_{\text{init}} = \mathbf{K}_2^T \mathbf{F} \mathbf{K}_1. \tag{2.23}$$

According to the properties of essential matrices [47], a non-zero matrix $\mathbf{E}$ is an essential matrix if and only if it has a singular value decomposition (SVD) $\mathbf{E} = \mathbf{U}\Sigma\mathbf{V}^T$ with:

$$\Sigma = \operatorname{diag}\{\sigma, \sigma, 0\} \tag{2.24}$$

for some $\sigma > 0$ and $\mathbf{U}, \mathbf{V} \in SO(3)$, i.e. a 3D rotation group [48]. The symbol "diag" represents diagonal matrix. To recover the final essential essential matrix $\mathbf{E}_{\text{fin}}$ and then the camera pose $\mathbf{T}$, we need to project $\mathbf{E}_{\text{init}}$ into the essential space by first computing its initial SVD:

$$\mathbf{E}_{\text{init}} = \mathbf{U}\Sigma_{\text{init}}\mathbf{V}^T \tag{2.25}$$

with $\Sigma_{\text{init}} = \operatorname{diag}\{\sigma_1, \sigma_2, \sigma_3\}$. Then we modify the matrix Σ_{init} to its final version Σ_{fin} that accords with the essential space's property: $\Sigma_{\text{fin}} = \operatorname{diag}\{\sigma, \sigma, 0\}$, where $\sigma = (\sigma_1 + \sigma_2)/2$. Hence, the final essential matrix is calculated as:

$$\mathbf{E}_{\text{fin}} = \mathbf{U}\Sigma_{\text{fin}}\mathbf{V}^T. \tag{2.26}$$

For $\mathbf{E}_{\text{fin}}$, there exist two possible solutions of the transformation:

$$\hat{\mathbf{t}}_1 = \mathbf{U}\mathbf{R}_z(+\frac{\pi}{2})\Sigma_{\text{fin}}\mathbf{U}^T,\ \mathbf{R}_1 = \mathbf{U}\mathbf{R}_z(+\frac{\pi}{2})\Sigma_{\text{fin}}\mathbf{V}^T; \tag{2.27}$$

$$\hat{\mathbf{t}}_2 = \mathbf{U}\mathbf{R}_z(-\frac{\pi}{2})\Sigma_{\text{fin}}\mathbf{U}^T,\ \mathbf{R}_2 = \mathbf{U}\mathbf{R}_z(-\frac{\pi}{2})\Sigma_{\text{fin}}\mathbf{V}^T, \tag{2.28}$$

where $\mathbf{R}_z$ represents the rotation matrix around the z–axis in a 3D Cartesian coordinate system, as is defined in Eq. (2.2). For $-\mathbf{E}_{\text{fin}}$, we have another two solutions. In the case of our AR-HUD calibration, we triangulate 2D feature point correspondences with these four candidate solutions and select a single solution which leads to the most positive depth values. The reason is that all the reconstructed 3D points should, in principle, lie in front of the windshield. This condition is called the cheirality constraint [49].

2.7 Scale-Invariant Feature Transform

As a classical algorithm in machine vision, SIFT was invented to detect and depict local features in images [50], [51]. Later in the target-free calibration scheme in Section 4.5.3, we apply it to find matching feature points between the pictures taken by our 3D sensor and the calibration camera. The following is an outline of the procedures.

Suppose we have a set of reference images I_{ref}, we first process them to extract the feature points, namely SIFT keypoints. We calculate the convolution results of these images using Gaussian filters at various scales k. This means, for an image $I(x, y)$ and a Gaussian filter $G(x, y, k\sigma)$ with

$$G(x, y, k\sigma) = \frac{1}{2\pi (k\sigma)^2} e^{-\frac{x^2+y^2}{2(k\sigma)^2}} \quad , \tag{2.29}$$

we do a convolution:

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) . \tag{2.30}$$

Because we have various scale factor k, we can obtain the Difference of Gaussians (DoG) image as:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) . \tag{2.31}$$

Next, we compare each pixel in $D(x, y, \sigma)$ with its 8 neighboring ones at the scale $k_i\sigma$ and 9 neighboring ones in neighboring scaled DoG images (scale k_{i+1}). Thus, we find the minima and maxima on these $D(x, y, \sigma)$ across scales and accordingly, extract candidate keypoints in the original reference images I_{ref}. A further fine selection of stable keypoints from these candidates includes interpolation, filtering low-contrast outliers, and rejecting edge responses.

The last step is to generate a descriptor for SIFT keypoints. A descriptor is a distinct vector to identify individual keypoints robust to adverse lighting conditions, changing views, and noise. It should also be able to match fast others belonging to the same features, so the vector size should be as small as possible. To this end, a weighted Gaussian window is defined around the keypoint, expanding to a 4×4 array of “neighbors”. Each neighbor covers 4×4 pixels. Gradient magnitudes and orientations are sampled within this window. The orientation is particularly classified into one of 8 bins summing up to 360°. Thus, the descriptor includes $4 \times 4 \times 8 = 128$ elements (4×4 is the number of neighbors, not pixels per neighbor). Finally, the descriptor is normalized to unit length, modified by thresholding the elements, and renormalized. Till now, the keypoint has a descriptor that is scale- and rotation-invariant, and robust against adverse illumination and noise.

2.8 Bilinear & Bicubic Interpolations

Bilinear and bicubic interpolations are widely used tools to reconstruct unknown information on a 2D area [52], [53]. Here we briefly introduce their mathematical models and solutions by taking a simple example. Assuming that we have a rectangular region of interest (ROI) on an image, whose corners are denoted as $p_{11} = [x_1, y_1]$, $p_{12} = [x_1, y_2]$, $p_{21} = [x_2, y_1]$ and $p_{22} = [x_2, y_2]$. We name these corners as sample points: the information on these corners (i.e. $f(p_{11})$, $f(p_{12})$, $f(p_{21})$ and $f(p_{22})$) is available, but that inside the rectangle is missing. Our objective is to interpolate the missing values inside the ROI based on the known boundary condition.

To retrieve the missing information within this ROI, we look for a function f as an approximation. For bilinear interpolation, the function is written as:

$$f(x, y) \approx a_0 + a_1 x + a_2 y + a_3 xy \ , \tag{2.32}$$

where $a_i, i \in \{0, 1, 2, 3\}$ are coefficients. The final solution is:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix}^{-1} \begin{bmatrix} f(p_{11}) \\ f(p_{12}) \\ f(p_{21}) \\ f(p_{22}) \end{bmatrix} \ . \tag{2.33}$$

For bicubic interpolation, the function is written as:

$$f(x, y) \approx \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j \ , \tag{2.34}$$

where we have 16 coefficients a_{ij}. The final solution is:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} f_{00} & f_{01} & f_{02} & f_{03} \\ f_{10} & f_{11} & f_{12} & f_{13} \\ f_{20} & f_{21} & f_{22} & f_{23} \\ f_{30} & f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} , \tag{2.35}$$

where

$$\begin{bmatrix} f_{00} & f_{01} & f_{02} & f_{03} \\ f_{10} & f_{11} & f_{12} & f_{13} \\ f_{20} & f_{21} & f_{22} & f_{23} \\ f_{30} & f_{31} & f_{32} & f_{33} \end{bmatrix} = \begin{bmatrix} f(p_{11}) & f(p_{12}) & f_y(p_{11}) & f_y(p_{12}) \\ f(p_{21}) & f(p_{22}) & f_y(p_{21}) & f_y(p_{22}) \\ f_x(p_{11}) & f_x(p_{12}) & f_{xy}(p_{11}) & f_{xy}(p_{12}) \\ f_x(p_{21}) & f_x(p_{22}) & f_{xy}(p_{21}) & f_{xy}(p_{22}) \end{bmatrix} \ . \tag{2.36}$$

Here, f_x, f_y represent the derivatives, while f_{xy} denotes the mixed partial derivative. To reconstruct an image of a fixed resolution, the more sample points we have, the more accurate this bicubic interpolation can be.

Later in our automatic AR-HUD calibration schemes and corresponding evaluation phases, we apply the bicubic interpolation to recover smooth warping maps for distortion correction. Note that we do not store the solved coefficients a_{ij}, but only warping maps themselves. Therefore, using warping maps is regarded as a nonparametric distortion compensation approach, as is detailed in Section 2.9.2. We exploit bilinear interpolation to recover both unknown linear projection parameters and warping maps at non-participating viewpoints.

2.9 Distortion Models

There are several different optical aberrations in automotive AR-HUDs, including the double image effect, astigmatism and distortion [54]. Other aberrations, e.g. chromatic aberration, spherical aberration and coma, have no or little impact. The double imaging comes from the mismatching reflections on the windshield's inner and outer surfaces, which will be detailed in Section 3.5. Astigmatism occurs due to the heterogeneous focal lengths in the horizontal and vertical directions, originating from both the HUD optics and the windshield. The most significant impact on the image quality comes from the optical distortion, which results from the same two sources as astigmatism. Though the current HUDs on the market usually have a narrow FOV, e.g. $8° \times 3°$ [55], the distortion is still observed. Note that the windshield's appearance is primarily designed not for HUDs but aerodynamic, safety, and even aesthetic goals. Hence, people try to match the HUD's reflective mirrors with the windshield to suppress the distortion. Nevertheless, for AR-HUDs, the residual distortion may still damage the projected image since they have a larger FOV and are much more sensitive to 2D–3D mappings than the non-AR ones. Moreover, it is impractical to decouple the HUD optics and the windshield in the calibration. Therefore, in previous work, the distortion correction is based on these two components as a whole.

Next, we introduce two relevant distortion models that have been included in our published work [24], [27], [56], [57]. The first is the conventional camera lens distortion model. It is a parametric model that can be recognized as a combination of radial, tangential, and thin prism distortion. This model's advantage is that it has been investigated in much previous work about camera calibration [13], [28], [58]. The second model is the nonparametric warping map, a 2D distribution describing pixels' warping on the virtual image. Both the models are view-dependent in our case. Accordingly, we have to consider their performances at the calibrated viewpoints inside the eye box and their extensibilities at non-participating viewpoints. In addition, there are some other distortion models in state of the art [10], [15], [18]. A comprehensive comparison and discussion is seen later in Section 6.4.

2.9.1 Camera Distortion Model

The conventional camera distortion model usually deals with the deformation of images caused by the imperfectly manufactured lenses with residual heterogeneity. In this case, an unobservable ground truth pixel $p_{\mathrm{gt}} = [u_{\mathrm{gt}}, v_{\mathrm{gt}}]^T$ moves to an observable distorted pixel

$p_{\text{dist}} = [u_{\text{dist}}, v_{\text{dist}}]^T$. Suppose that we have a distortion center $p_{\text{dc}} = [u_{\text{dc}}, v_{\text{dc}}]^T$ on the image, this model considers three terms [28], [59]:

- radial distortion caused by defective radial curvature of a lens;
- tangential distortion arising from decentering of the lens and other optical components;
- thin prism distortion coming from a lens' tilt w.r.t. the sensor's pixel plane.

These distortions can be expressed using the following equations:

$$\begin{cases} u_{\text{dist}} &= u_{\text{gt}} + \bar{u}_{\text{gt}}(k_1 r_{\text{gt}}^2 + k_2 r_{\text{gt}}^4) + \left[p_1(r_{\text{gt}}^2 + 2\bar{u}_{\text{gt}}^2) + 2p_2\bar{u}_{\text{gt}}\bar{v}_{\text{gt}}\right] + s_1 r_{\text{gt}}^2 \\ v_{\text{dist}} &= v_{\text{gt}} + \bar{v}_{\text{gt}}(k_1 r_{\text{gt}}^2 + k_2 r_{\text{gt}}^4) + \left[p_2(r_{\text{gt}}^2 + 2\bar{v}_{\text{gt}}^2) + 2p_1\bar{u}_{\text{gt}}\bar{v}_{\text{gt}}\right] + s_2 r_{\text{gt}}^2 \end{cases}, \tag{2.37}$$

where $\bar{u}_{\text{gt}} = u_{\text{gt}} - u_{\text{dc}}$, $\bar{v}_{\text{gt}} = v_{\text{gt}} - v_{\text{dc}}$, $r_{\text{gt}} = \sqrt{\bar{u}_{\text{gt}}^2 + \bar{v}_{\text{gt}}^2}$. The coefficents k_i, p_i and s_i $(i \in \{1, 2\})$ represent the radial, tangential and thin prism distortions, respectively. Figure 2.10 illustrates typical examples of radial lens distortion, i.e. barrel and pincushion distortions. Because p_{gt} is undetectable target values, a pair of inversed equations are often used, where p_{gt} and p_{dist} are interchanged:

$$\begin{cases} u_{\text{gt}} &= u_{\text{dist}} + \bar{u}_{\text{dist}}(k_1 r_{\text{dist}}^2 + k_2 r_{\text{dist}}^4) + \left[p_1(r_{\text{dist}}^2 + 2\bar{u}_{\text{dist}}^2) + 2p_2\bar{u}_{\text{dist}}\bar{v}_{\text{dist}}\right] + s_1 r_{\text{dist}}^2 \\ v_{\text{gt}} &= v_{\text{dist}} + \bar{v}_{\text{dist}}(k_1 r_{\text{dist}}^2 + k_2 r_{\text{dist}}^4) + \left[p_2(r_{\text{dist}}^2 + 2\bar{v}_{\text{dist}}^2) + 2p_1\bar{u}_{\text{dist}}\bar{v}_{\text{dist}}\right] + s_2 r_{\text{dist}}^2 \end{cases}, \tag{2.38}$$

where $\bar{u}_{\text{dist}} = u_{\text{dist}} - u_{dc}$, $\bar{v}_{\text{dist}} = v_{\text{dist}} - v_{dc}$, $r_{\text{dist}} = \sqrt{\bar{u}_{\text{dist}}^2 + \bar{v}_{\text{dist}}^2}$. For simplicity in calculation, we employ a slightly variant form coupling the tangential and thin prism distortion terms together [28]:

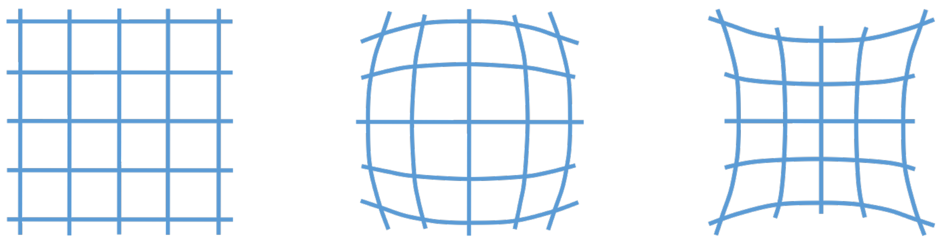

(a) Normal object. (b) Barrel. (c) Pincushion.

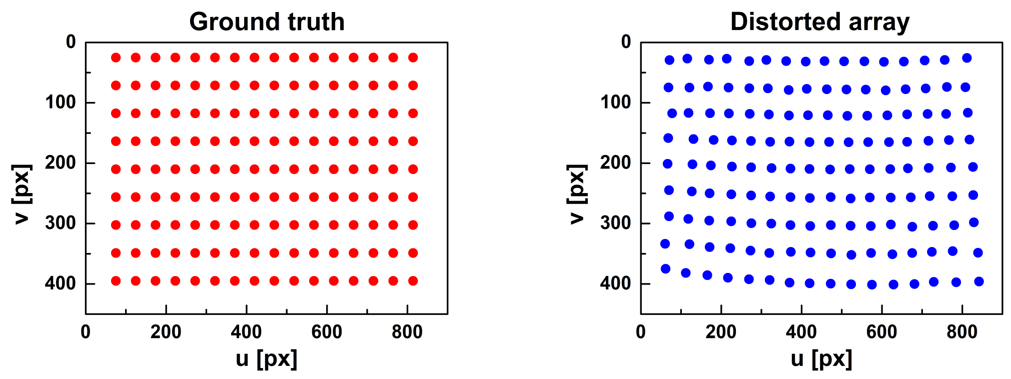
Figure 2.10: Typical examples of radial camera lens distortion, including the barrel (b) and pincushion (c) distortions.

$$\begin{cases} u_{\text{gt}} &= u_{\text{dist}} + k_1\bar{u}_{\text{dist}} r_{\text{dist}}^2 + k_2\bar{u}_{\text{dist}} r_{\text{dist}}^4 + q_1 r_{\text{dist}}^2 + \bar{u}_{\text{dist}}(p_1\bar{u}_{\text{dist}} + p_2\bar{v}_{\text{dist}}) \\ v_{\text{gt}} &= v_{\text{dist}} + k_1\bar{v}_{\text{dist}} r_{\text{dist}}^2 + k_2\bar{v}_{\text{dist}} r_{\text{dist}}^4 + q_2 r_{\text{dist}}^2 + \bar{v}_{\text{dist}}(p_1\bar{u}_{\text{dist}} + p_2\bar{v}_{\text{dist}}), \end{cases} \tag{2.39}$$

where $q_i = p_i + s_i$ for $i \in \{1, 2\}$. Though this model is initially raised to deal with camera lens distortion, we still have found its effectiveness for the HUD case at individual viewpoints. The reason behind this is that a curved windshield can be regarded as a refractive element like an irregularly formed lens in the virtual optical path from the virtual image to the viewpoint [27].

2.9.2 Warping Maps

Warping an image means the operation of rearranging its pixels so that the shapes of projected patterns change. An example consisting of a point array is illustrated in Figure 2.11. Hence, non-chromatic optical aberrations can be recognized as different sorts of warping. Another example of image warping is the earlier mentioned camera lens distortion or the opposite operation to restore objects' artifact-free forms. To tell the difference, we regard the optical distortion as a forward warping and the correction as a backward warping or de-warping. Indeed, the camera distortion model in Eq. (2.37) provides a parametric forward warping, while that in Eq. (2.38) and (2.39) present a parametric backward warping.



(a) Ground truth array. (b) Distorted array.

Figure 2.11: A 9×16 rectangular ground truth point array (a), and its warped counterpart (b) deviating from a perfect rectangle.

Our warping maps, however, characterize nonparametric 2D–2D correspondences between the source and target images. Each correspondence can be expressed by a bias vector:

$$\mathbf{v}_{\text{bias}} = [\Delta u, \Delta v]^T = [u_{\text{gt}} - u_{\text{dist}}, v_{\text{gt}} - v_{\text{dist}}]^T . \tag{2.40}$$

We plot a pair of simulated warping maps in Figure 2.12. Mathematically, a warping map may be two-dimensionally smooth, or there can exist non-differentiable pixels. Since the reflective mirrors inside the HUD projector and the windshield are recognized as smoothly formed, we assume that our warping maps are smooth binary functions with:

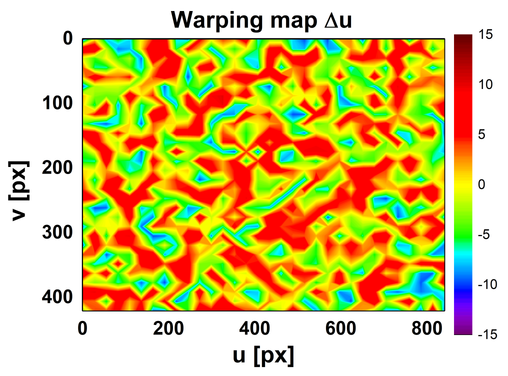
$$\begin{cases} w_{\Delta u} = f_{\Delta u}(u, v) \\ w_{\Delta v} = f_{\Delta v}(u, v) \end{cases} . \tag{2.41}$$

Alternatively, these relations can be expressed using two matrices:

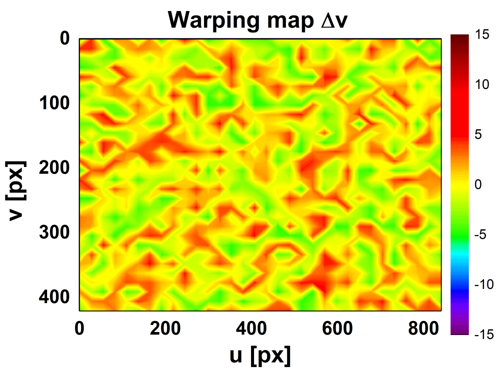
$$\begin{cases} w_{\Delta u} = \begin{bmatrix} \Delta u_{11} & \dots & \Delta u_{1n} \\ \vdots & \ddots & \vdots \\ \Delta u_{m1} & \dots & \Delta u_{mn} \end{bmatrix} \\ w_{\Delta v} = \begin{bmatrix} \Delta v_{11} & \dots & \Delta v_{1n} \\ \vdots & \ddots & \vdots \\ \Delta v_{m1} & \dots & \Delta v_{mn} \end{bmatrix} \end{cases}, \tag{2.42}$$

if we the image resolution is $n \times m$ pixels. Note that under our definition, a warping map is similar to specific physics concepts, like an electric or magnetic field, so that it can be superimposed with others to form new ones. Hence, we can try using some interpolation or regression functions to reconstruct an entire warping map from several discrete known bias vectors $\mathbf{v}_{\text{bias}}$. Of course, a broader sampling of these vectors generally helps in obtaining a more precise mapping. Later in Section 5.5, we will show some examples of warping maps that we have acquired in practical calibrations.

Here we also point out that warping is dependent on the selected viewpoint V_i because the distortion varies with the optical path, as is captured in Figure 2.13. Put alternatively, when the optical path from the HUD projector to the image receiver changes, the reflective area (or rather, the reflection points in Figure 2.1) on the windshield's inner surface moves. Considering varying viewpoints, using warping maps is a reasonable option to deal with the optical distortion because physically, it should change continuously with the viewpoint position. Nevertheless, this approach faces the problem of an extensive volume of data. For example, if we have a virtual image of a resolution of 1024×768, then the size of a single warping map is nearly 800 thousand pixels. It might bring a challenge to the computational and storage capacity for on-vehicle computers.



(a) Warping map in u–direction.



(b) Warping map in v–direction.

Figure 2.12: A pair of simulated warping maps presenting the Δu– (a) and Δv–components (b) of bias vectors, plotted as heat maps. The value at a pixel $[u, v]^T$ equals Δu or Δv in the corresponding bias vector $\mathbf{v}_{\text{bias}}$. px: pixel.

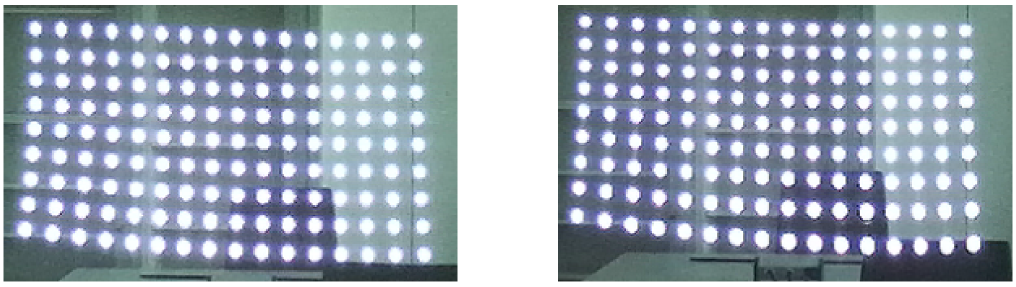

(a) Virtual point array at the first viewpoint. (b) Virtual point array at the second viewpoint.

Figure 2.13: View-dependent optical distortions in the rendering via our AR-HUD. Here we display the same 9×16 rectangular virtual point array and take pictures at two separate viewpoints. We notice that the arrays are both distorted but in slightly different manners.

2.10 Levenberg–Marquardt Algorithm

Levenberg–Marquardt algorithm is an iterative nonlinear optimization tool that is intensively used in camera calibration to refine solutions [13], [60], [61]. Generally speaking, it is one of the applicable techniques to deal with the least-square problem.

Let us take an uncomplicated example about curve fitting here. Suppose we have n observed pairs of data $[x_i, y_i]$, where x_i are independent and y_i are dependent variables. The task is to solve the least-square problem in the following form:

$$\hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) \equiv \arg\min_{\boldsymbol{\theta}} \frac{1}{2} \sum_{i=1}^{n} [y_i - f(x_i, \boldsymbol{\theta})]^2 , \tag{2.43}$$

where $f(x, \boldsymbol{\theta})$ is the model curve function with a set of parameters $\boldsymbol{\theta}$ and $F(\boldsymbol{\theta})$ denotes the sum of deviations' squares.

To run iterations for solving $\boldsymbol{\theta}$, we should carefully choose an initial guess to achieve the global minimum; otherwise, the iteration stops at a local minimum. We update the parameter set $\boldsymbol{\theta}$ in each iteration to a new set $\boldsymbol{\theta} + \mathbf{h}$. Hence, we approximate an updated function using the Taylor series:

$$f(x_i, \boldsymbol{\theta} + \mathbf{h}) \approx f(x_i, \boldsymbol{\theta}) + \mathbf{J}_i \mathbf{h} , \tag{2.44}$$

where

$$\mathbf{J}_i = \frac{\partial f(x_i, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \tag{2.45}$$

expresses the gradient of our model w.r.t. $\boldsymbol{\theta}$. We convert it into a vector form:

$$\mathbf{f}(\boldsymbol{\theta} + \mathbf{h}) \approx \mathbf{f}(\boldsymbol{\theta}) + \mathbf{J}\mathbf{h} , \tag{2.46}$$

where $\mathbf{J}$ is the Jacobian matrix. Then we can approximate:

$$\begin{aligned} F(\boldsymbol{\theta}+\mathbf{h}) &= \frac{1}{2}(\mathbf{y}-\mathbf{f}-\mathbf{J}\mathbf{h})^T(\mathbf{y}-\mathbf{f}-\mathbf{J}\mathbf{h}) \\ &= \frac{1}{2}\left[(\mathbf{y}-\mathbf{f})^T(\mathbf{y}-\mathbf{f})-(\mathbf{y}-\mathbf{f})^T\mathbf{J}\mathbf{h}+\mathbf{h}^T\mathbf{J}^T\mathbf{J}\mathbf{h}\right] . \end{aligned} \tag{2.47}$$

Here, we introduce a non-negative damping parameter μ and solve

$$\mathbf{h} = \arg\min_{\mathbf{h}}\left[F(\boldsymbol{\theta}+\mathbf{h})+\frac{1}{2}\mu\mathbf{h}^T\mathbf{h}\right] \tag{2.48}$$

by forcing its partial derivative w.r.t. $\mathbf{h}$ equal to zero:

$$\mathbf{J}^T(\mathbf{y}-\mathbf{f})-\mathbf{J}^T\mathbf{J}\mathbf{h}-\mu\mathbf{h}=\mathbf{0}, \tag{2.49}$$

$$\left(\mathbf{J}^T\mathbf{J}+\mu\mathbf{I}\right)\mathbf{h}=(\mathbf{H}+\mu\mathbf{I})\mathbf{h}=\mathbf{J}^T(\mathbf{y}-\mathbf{f}), \tag{2.50}$$

where $\mathbf{I}$ is the identity matrix. The damping factor's functionalities include:

- to $\mu > 0$, the matrix $(\mathbf{H}+\mu\mathbf{I})$ is positive-definite;
- when μ is large, the case matches the gradient descent. The iteration converges fast when it is relatively far from the final solution;
- when μ is small, the case matches Gauss–Newton algorithm. It leads to a quadratic rate of convergence when the iteration is close to the final solution.

In short, Levenberg–Marquardt algorithm combines the strengths of gradient descent and Gauss–Newton algorithms. Later we use it to optimize our calibrated parameters (see also Section 4.5.2.4).

3. System

As is stated in Section 2.1, the AR rendering involves multiple coordinate systems, i.e. the 3D world (W), viewpoint (V), HUD-FOV (H) and driver camera (D) spaces, as well as the 2D virtual image space (I). The parametric part of calibration includes determining the transformations between the above 3D reference frames and the 3D–2D perspective projection. Since the viewpoint space, the HUD-FOV space, and the virtual image vary with the viewpoint position, the calibration has to consider multiple viewpoints. These coordinate systems are, indeed, specified by the pose of sensors in actual experiments.

Therefore, this chapter describes our AR-HUD system, especially those involving hardware devices with their connections and the software modules. Some hardware devices are the same as in vehicles, while others are for setups in our laboratory environment. They include multiple real and virtual sensors, and reflective interfaces. Section 3.1 introduces our 3D sensor, i.e. a stereo camera for detecting calibration targets or environmental graphic features. The following Section 3.2 details the selection of eye box and the mounting for our calibration cameras. Next, we focus on the AR-HUD projector and its image generation principles in Section 3.3, while Section 3.4 introduces the virtual and real calibration cameras that observe virtual images at viewpoints. Section 3.5 presents the windshield for an AR-HUD system and its characteristics for image reflection. We depict the driver camera in Section 3.6, which tracks the viewpoint live. Lastly, we illustrate the software modules in Section 3.7, which are responsible for controlling hardware devices, detection and data processing.

3.1 3D Sensor

For an AR-HUD, we need an extra 3D sensor that provides information about the real-time forward scene because we want virtual objects to overlap with the real ones. Such is the immersive effect in the driver's view, as is introduced in Section 1.1. As is illustrated in Figure 2.1, the 3D sensor is usually embedded behind the windscreen or fixed on top of the vehicle. It detects real objects in the frontal environment and returns their positions in the world space W. Without loss of generality, it can be a stereo camera, a LiDAR, an RGB-D camera or other devices with an adequate 3D sensing capability.

Figure 3.1 shows a stereo camera (Stereolabs ZED) that we exploit as the 3D sensor. It is a passive optical sensor that only receives but does not emit photons. The stereo camera works in the way of human eyes based on the stereo vision principles in Section 2.5. It has two identically designed monocular cameras whose optical axes are oriented in parallel and whose optical centers are separated horizontally by a baseline distance $B_{\mathrm{ZED}} = 120\,\mathrm{mm}$. When a 3D position P to augment is detected, its projection p_i in the virtual image I_i is calculated using the calibrated or interpolated parameters relating to the current viewpoint V_i. Then the corresponding pixel in the PGU is "switched on". Thereby, the automotive AR-HUD has augmented the real-world objects with virtual ones. We also employ a second ZED stereo camera as the driver camera, as is detailed in Section 3.6.

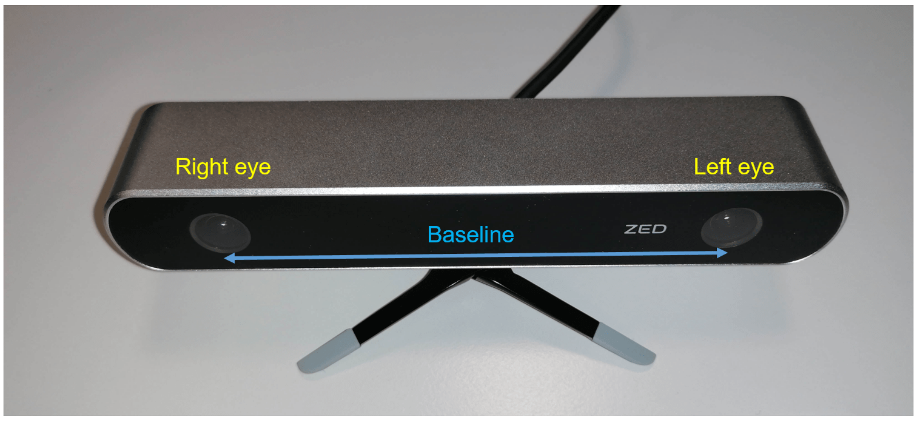
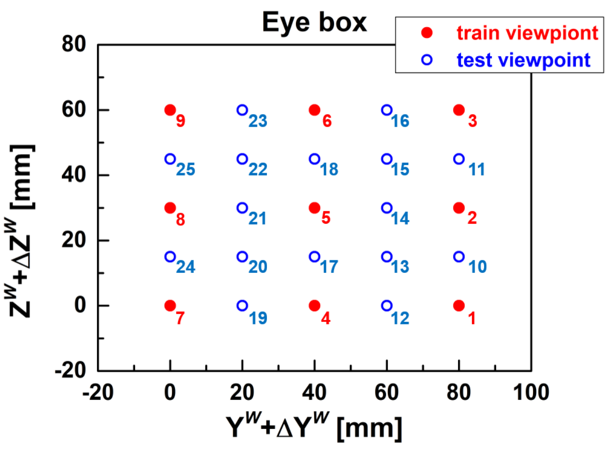


Figure 3.1: A ZED stereo camera that serves as our 3D sensor in the AR-HUD system. Another such camera serves as the driver camera. Note that the left and right eyes are marked inversely because the two mono-cameras are facing out.

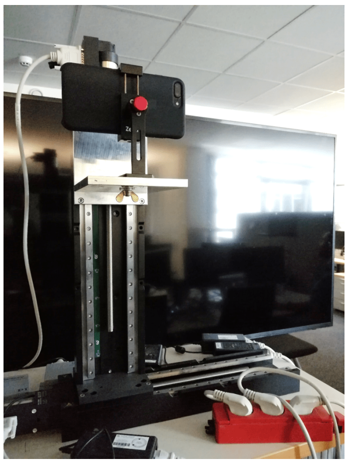
3.2 Eye Box

Theoretically, we have to enumerate all possible viewpoints to ensure the completeness of calibration considering drivers' various heights and head motions. Both the height distribution and the head motion appear randomly within reasonable intervals. However, it is impractical to include all these viewpoints because there are countless ones. Therefore, our strategy is to define an "eye box" in which the viewpoints frequently appear. A cubic eye box is sketched in Figure 2.3 as the example. Nonetheless, the viewpoint sampling inside the eye box is also crucial for calibration implementation. Suppose that we have an eye box of 100 mm × 100 mm × 100 mm, if our sampling step is 5 mm on each orthogonal axis, then we have $21^3 = 9261$ different viewpoints. In this case, even if the calibration at a single viewpoint takes only 5 s, a complete calibration will take more than half a day. This time consumption is intolerable in the automotive industry and especially for manual implementation, which will bring too massive a burden to human labor. Therefore, we have to sample fewer viewpoints and develop a robust estimation for calibration parameters at other uninvolved ones while sacrificing the accuracy or robustness as little as possible.

As a result, we select several viewpoints inside the eye box as the train set, at which we complete the calibration. For an uncomplicated scenario, these discrete viewpoints should be evenly distributed. Afterward, we can readily compute the calibration parameters at any non-participating viewpoint using interpolation. Practically, our eye box is a 2D rectangular area lying within the $Y^W Z^W$–plane. We ignore the displacement along X^W–axis because it has little effect on the final augmenting due to limited opening angles of the HUD-FOV [23], [24]. This empirical consideration is also validated by our simulation results in Section 5.2.5. An example of eye box is plotted in Figure 3.2 (a). The 80 mm × 60 mm rectangle includes 9 train and 16 test viewpoints. The train viewpoints are mainly used for initial calibration and validation (sometimes also for evaluation), while the test ones are employed for the evaluation phase, which will be detailed in Section 4.1.



(a) Eye box.



(b) Moving stage.

Figure 3.2: A 2D eye box (a) with 9 train (red circle) and 16 test (blue rings) viewpoints, and a moving stage (b) comprising two perpendicular linear axes along Y^W– and Z^W–directions, respectively. A smartphone is fixed on the stage, whose camera faces out. ΔY^W and ΔZ^W represent some offsets.

3.3 HUD Projector

In this part, we first demonstrate the structure of our AR-HUD projector box. Then, we state some particular requirements on the projector's design and the trade-offs. They are directly or indirectly related to our thoughts on the calibration. Moreover, we give a brief introduction of the popular applied display technology for AR-HUDs, i.e. digital light processing (DLP).

3.3.1 Structure

Figure 3.3 illustrates the basic schematic of an AR-HUD projector box, which comprises a light source, i.e. a PGU, some aspherical reflective mirrors, a glare trap and a light trap. Images are generated from the PGU and bounced by the mirrors. The projected light goes through the HUD box's glare trap and reaches the windshield's inner surface, followed by a final reflection towards the driver's eyes. Notice that when the driver's pose is unchanged, the optical path from each pixel to the driver's single eye is fixed. Regardless of the secondary reflection from the windshield's outer surface, each optical ray corresponds to a reflection point onside the windshield's inner surface, as is already marked in Figure 2.1. The light trap is deployed close to the glare trap, serving as the exiting window for the reflected sunlight.

3.3.2 Requirement & Compromise

The HUD projector is the hardware module where images are generated and projected upward to the windshield. Non-AR-HUD projectors are generally featured a relatively compact volume (e.g. 5 L), a narrow FOV (e.g. 8° horizontal and 3° vertical) and short focal lengths (e.g. 2 m) [55], because they only provide drivers static information, as is introduced in Section 1.1. On the contrary, since an AR-HUD projector transmits images

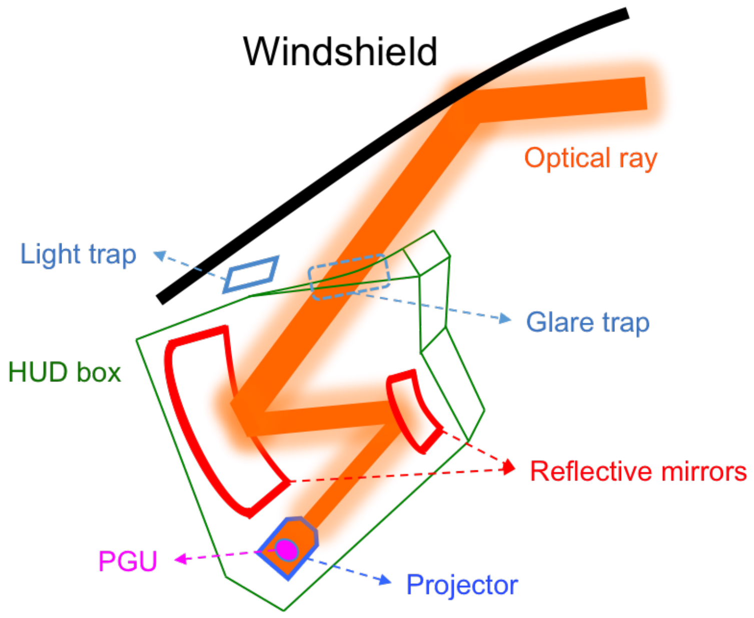


Figure 3.3: Structure of an automotive AR-HUD projector box, which contains a light-source (PGU), aspherical mirrors, a glare trap and a light trap.

that are immersive in the real world, it should be designed with a more expanded FOV to cover the street scene, as well as a longer focal length (e.g. $\geq 10\,\mathrm{m}$) to make virtual objects closer to real ones, which reduces the accommodation requirement on drivers' eyes. Indeed, the larger its FOV is, the more useful immersive information it can deliver and the higher replacement ability the AR-HUD has versus the dashboard screen. According to some investigation, to fulfill the rendering of virtual objects in both urban and highway traffic, an AR-HUD should have a FOV broader than $20^\circ \times 20^\circ$ and a minimum focal length of $30\,\mathrm{m}$ [55]. Nonetheless, to our best knowledge, automotive AR-HUDs with such a large FOV and a focal length are still under development.

Therefore, the systematical designs for AR-HUDs are much harsher than for non-AR-HUDs. There are more technical requirements than for non-AR-HUDs. For example, since the projector box is embedded in the car's dash, a trade-off appears between its volume and the FOV: an AR-HUD projector can occupy up to $20\,\mathrm{L}$ [55]. Such a much larger volume brings a challenge for organizing the available space in the dash. Meanwhile, people should concern additional issues, such as power consumption, heat management, costs, solar load, and mechanical stability. Though some new types of AR-HUDs, e.g. holographic waveguide-based AR-HUDs, can save lots of space, they are still, at this moment, not yet available for mass production or extensive use. The larger volume of an AR-HUD also brings more requirements for the windshield. For example, in some AR-HUD systems, there is a film filter or a wedge layer [62] inserted between the windshield glass layers (see also Section 3.5). Such a mechanism is employed to suppress the secondary optical reflection from the windshield's outer surface. This intermediary layer has to be sizable enough to cover the entire FOV, bringing extra windshield production costs.

3.3.3 Digital Light Processing

Presently, the most prevalent display technology for AR-HUDs is the DLP. This technology's core is its microelectromechanical system (MEMS) device, i.e. a dense semiconductor-based array of fast responding reflective digital light switches. Therefore, this MEMS chipset is also named as digital micromirror device (DMD), whose pitch size is usually not larger than 5.4 µm.[1] As is illustrated in Figure 3.4, the homogeneous unmodulated incident light is digitally modulated on the reflective DMD and then projected to the user via later-stage optics. Alternatively speaking, the modulation occurs not directly in the initial light source but in the semiconductor-based optical reflector. Only after this modulation, the projected light carries information for the driver. The micromirrors can be rapidly turned on or off (e.g. 60 Hz), which follow the on-off ratio in the electrical modulation signal. Usually, the micromirror array size adapts to the resolution of projected images, such as 800×600, 1024×768, or even 1920×1080 (Full HD) pixels.

Projectors based on DLP have demonstrated many advantages compared to those adopting other technologies, e.g. liquid-crystal display (LCD) [63]:

- Since the modulation happens on the DMD, both the optical efficiency at pixels and their brightness are enhanced;
- Because DLP is based on the reflection from the DMD, whose chipset can be cooled rather uncomplicatedly by the help of a substrate, it supports the use of bright illumination sources, such as high-power lamps or lasers;
- A DLP projector is a bidirectional digital device that receives the input signal (modulation) and returns the output signal both in digital form, which guarantees the image quality and stability.

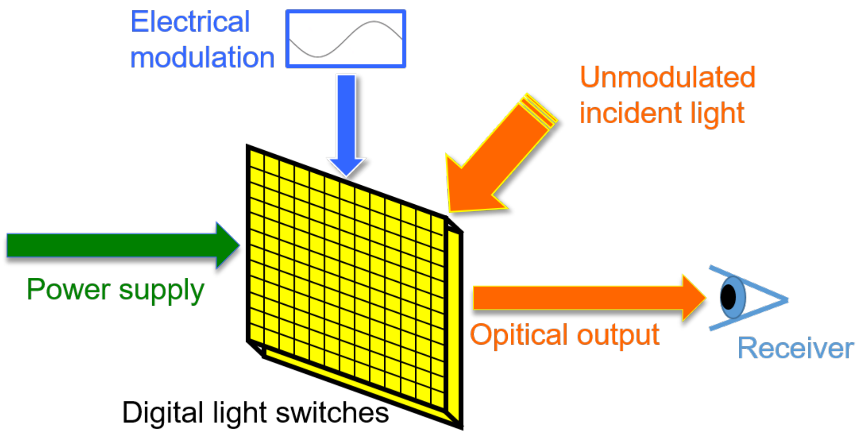


Figure 3.4: Schematic of a DLP array. The homogeneous unmodulated incident light is modulated on thousands of micromirrors by the digital input electrical signals before its digital output is projected to the receiver.

[1] `https://www.ti.com/product/DLP3310`, Retrieved 2020-05-24.

3.4 Monocular Calibration Cameras

In our calibration, we can place a virtual or real calibration camera at the viewpoint to observe the target, frontal scene and virtual images. It collects 2D information in the virtual image space I or helps reconstruct 3D geometry in the world space W.

3.4.1 Virtual Calibration Camera

The former option is named virtual because it is, indeed, one of the operator's eyes. Note the left and right eyes have an overlapping binocular FOV, as is illustrated in Figure 3.5. In the manual calibration approach with human eyes (see Section 4.4.1.1), we calibrate our AR-HUDs using one of the two pupils while the other maintains closed or blocked. The purpose is to avoid double perceptions in human stereo vision when the operator observes the virtual image. Hence, the virtual camera in our calibration is kept monocular. However, drivers' both eyes are open when they are driving. Therefore in the AR rendering using a calibrated AR-HUD, the middle point between the driver's pupils is recognized as the viewpoint, as is interpreted in Section 2.4.

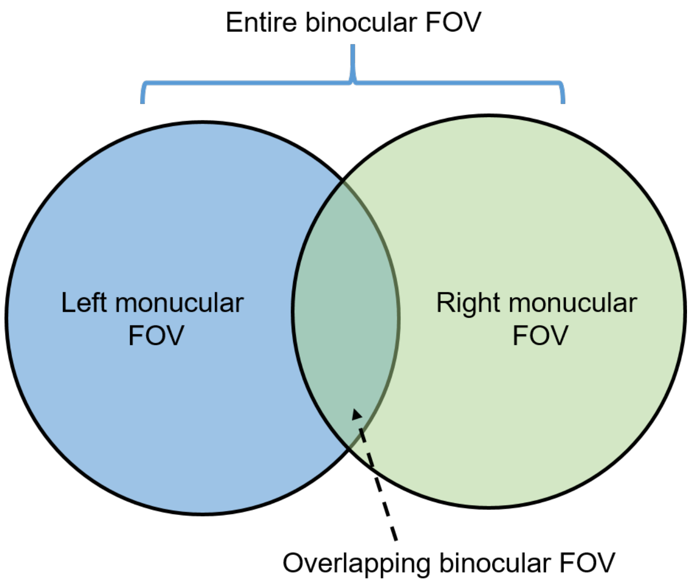


Figure 3.5: Schematic of human binocular vision, including two individual monocular FOVs and their overlapping FOV.

3.4.2 Real Calibration Camera

At the viewpoint, a real monocular calibration camera collects real-time images or videos so that we can reduce the human labor in the manual calibration or directly apply various fast image processing techniques in the automatic calibration pipeline. It can be either a monocular camera [10], [16], [24], [27] or a stereo camera [18]. There is, however, a few requirements in selecting the calibration camera. First of all, it should have enough high resolution, e.g. Full HD resolution, otherwise the feature points on the target (e.g. chessboard corners) or in the forward scene (e.g. inherent keypoints), or rendered virtual points can be unclear in the captured pictures, which may bring inconvenience for the image processing at later stages. Secondly, it is recommended that the camera is connected (e.g. wireless via a virtual IP address or via a cable) to a computer so that the photos or clips are readily accessible. This can significantly enhance the efficiency of running the experiment; otherwise, the camera might have to be removed from the mounting and then carefully repositioned after we derive the data. Thirdly, those cameras with strong lens distortion should be excluded from selection, e.g. fish-eye cameras, since they require extra efforts in rectifying the severely distorted images.

Therefore, we use a consumer-grade smartphone camera (Huawei P10 Lite or Apple iPhone 7 Plus) to take photos at viewpoints. The camera is connected to a computer (Lenovo P50) wirelessly via some webcam apps[2] [3] so that we can observe the captured live photos and video stream. We mount the camera on a moving stage consisting of two motorized linear axes (Physik Instrumente M-414) to sweep different viewpoints. The stage, as is shown in Figure 3.2 (b), can move both fast (100 mm/s) and precisely (0.5 µm resolution) under home-made controlling program. In the preparation phase of calibration, we will first pre-calibrate the smartphone camera to obtain its intrinsic matrix (like $\mathbf{K}$ in Eq. (2.8)) and distortion coefficients, and use them to rectify the captured photos. We also utilize these data to estimate the intrinsic and extrinsic parameters of our AR-HUD.

3.5 Windshield

The windshield of a vehicle with HUD should be specially designed and manufactured to mitigate the harmful double reflection. As is shown in Figure 3.6 (a–b), when the HUD projects images upwards to the windshield, the first reflection happens on its inner surface, forming the chief virtual image; the second reflection comes from refraction into the windshield glass, a reflection on its outer surface, and then backward refraction towards out of the glass, forming the fuzzy "ghost" image [11]. Both reflections obey the laws in Section 2.3. On the one hand, an incident optical ray is split as two output rays, which do not superimpose each other spatially; on the other hand, the driver receives two separate rays from the same pixel in the PGU.

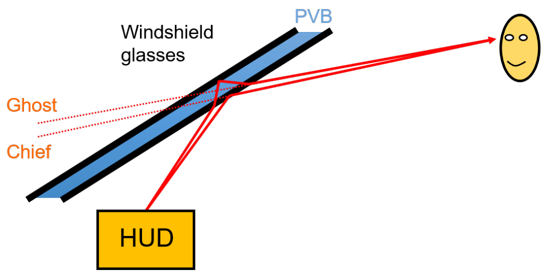
There are three prevalent solutions to this double reflection problem. The first is to attach a dark film in the windshield to block the second reflection partially. However, this approach faces the risk of lowering the windshield's transparency, resulting in safety issues for the drivers. The second is to use a wedge-formed intermediate layer [64], [65] or doublet glasses [66] to make the two reflected images close to each other (Figure 3.6 (c–d)), or to split the secondary image far beyond the first one. The last approach is to use a thin-film beam splitter [67] made of polyethylene propylene (PEP) substrate and specially coated with a multilayer, which should guarantee a nearly constant reflectance over the visible spectrum. This can mitigate the influence of double reflection while reaching a transmittance $\geq 70\%$ [62]. However, all these solutions bring extra costs for the design

[2] `https://play.google.com/store/apps/details?id=com.pas.webcam.pro&hl=de`, Retrieved 2020-05-24.

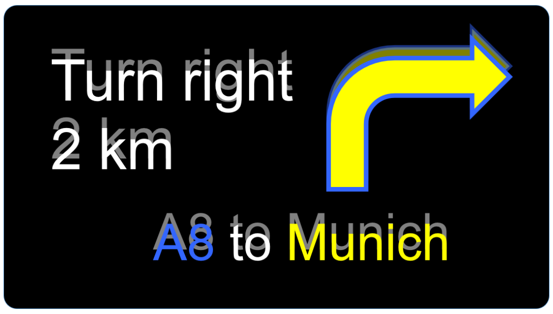
[3] `https://apps.apple.com/de/app/epoccam-webcam-for-mac-and-pc/id449133483`, Retrieved 2020-05-24.

and production of the windscreen. Particularly the second solution should perfectly match the geometric optical characteristics of the HUD projector.

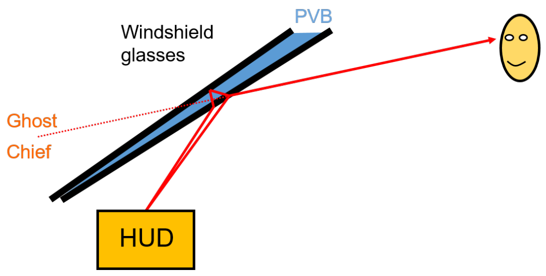
In our laboratory and on-vehicle experiments, the employed windshields are adaptively designed for our HUD projector box. Therefore, the second reflection is significantly suppressed, as is earlier demonstrated in Figure 2.13.



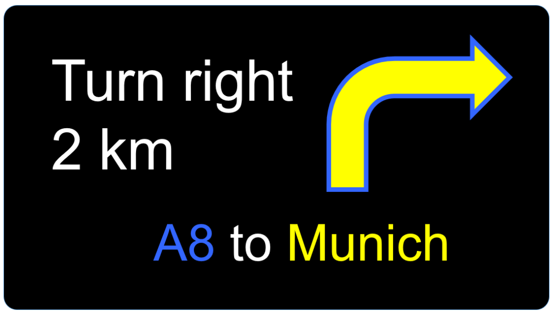
(a) Two optical rays from the same pixel in PGU reach driver's sight, forming the separate chief and ghost images.



(b) Illustration of unmatched chief and secondary images. Note that the latter is upward shifted compared to the former.



(c) A wedge angle in the windshield forces overlapping of optical rays.



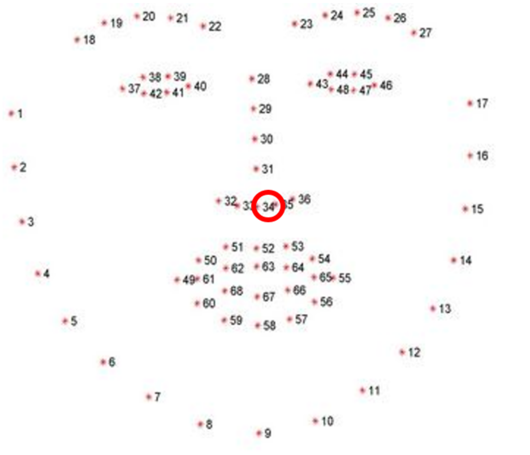
(d) Illustration of matched chief and secondary images.

Figure 3.6: Double reflection in an AR-HUD system (a–b), which occurs on the inner and outer surfaces of the windshield. It can be mitigated using a wedge-form windshield (c–d). PVB: polyvinyl butyral.

3.6 Driver Camera

An in-vehicle driver camera, or rather, head tracker, recognizes the human face and detects the driver's eye positions. It can be an RGB-D camera [68] or a stereo camera [69]. It has various applications in modern vehicles. An example is to monitor the driver's attention in case of drowsy driving. It can also help in the calibrations of the driver's seat, the rear-view and side mirrors, and the steering wheel. In principle, the driver camera has its own coordinate system, i.e. the driver camera space D mentioned in Section 2.1.5. If it is pre-aligned to the 3D sensor, it can also track the viewpoint in the world space W.

Throughout our pipeline, we use a second stereo camera (Stereolabs ZED) as the driver camera. We program it diversely so that in the manual AR-HUD calibration scheme with human eyes, it can recognize the operator's facial features and find the opening eye, which is shown in Figure 3.7; in the implementations with a real calibration camera, the driver camera can monitor its real-time position. In the consequent AR rendering, the driver camera first returns the real-time viewpoint to the computer. Then accordingly, view-related parameters, e.g. extrinsic matrix $\mathbf{T}_{H_iW}$ in Eq. (2.7), are substituted into the pinhole model (Eq. (2.5)) to project 3D positions in the world W to 2D pixels in the virtual image I_i, followed by a distortion correction step.



(a) Feature points on a face profile, with Point 34 circled.

(b) A human face picture on which Point 34 is recognized and marked red.

Figure 3.7: Face recognition by our programmed ZED stereo camera as the driver camera. There are 68 feature points[4] (a) detectable on a human face, one of which is marked on a human face photo (b).

However, if the driver camera is not accurately calibrated or aligned with the world space W, there exist biases in the acquired viewpoint position. Consequently, the calculated projection parameters are biased, which will lead to wrongly positioned virtual objects. Therefore, in Section 4.2, we will introduce a calibration method for the driver camera using the AR-HUD calibration results.

3.7 Software

Our software part consists of three sub-modules: controlling module, detection module and data processing module. Their specific duties are also detailed in Chapter 4, combining our concrete calibration schemes.

The controlling part is responsible for arranging related hardware and software components to run the calibration experiment. It organizes the initial calibration, validation, and evaluation phases in the entire calibration pipeline (see Section 4.1). Its tasks can be categorized as follows:

1. rendering calibration target pattern on an external screen;
2. changing the viewpoint position by controlling the moving stage for the calibration camera;

[4]Sub-figure (a) is adapted using the picture in *GitHub*: `https://github.com/raviranjan0309/Detect-Facial-Features`, Retrieved 2018-07-18.

3. rendering of virtual images via the HUD, including the boundary of HUD-FOV and virtual point arrays;
4. support manual and automatic shifting of virtual points to their corresponding 3D control points;
5. interacting with the operator.

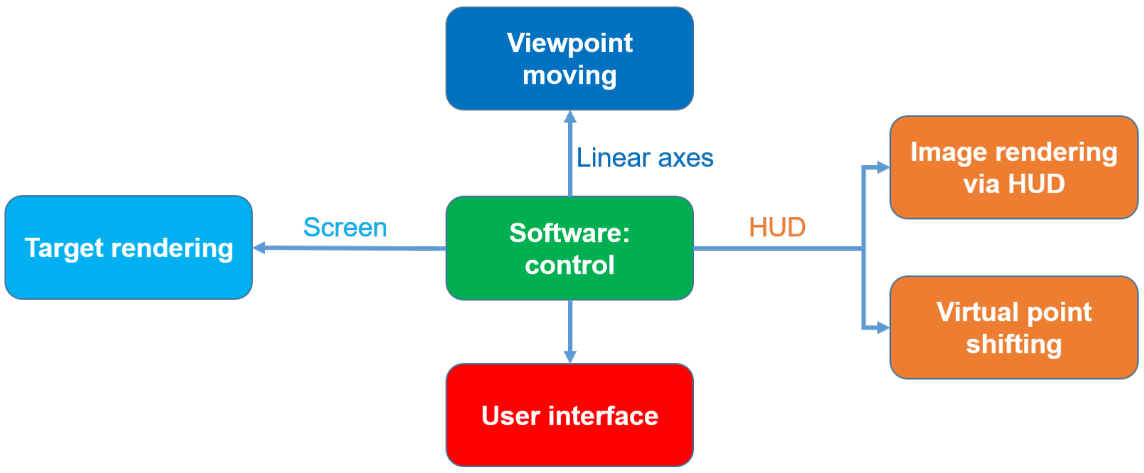
The detection part is responsible for various detection tasks. It supports:

1. target and feature point detection via our 1st ZED stereo camera (3D sensor);
2. viewpoint tracking via our 2nd ZED stereo camera (driver camera);
3. virtual point and HUD-FOV boundary detection via our smartphone camera (calibration camera);
4. interacting with the operator.

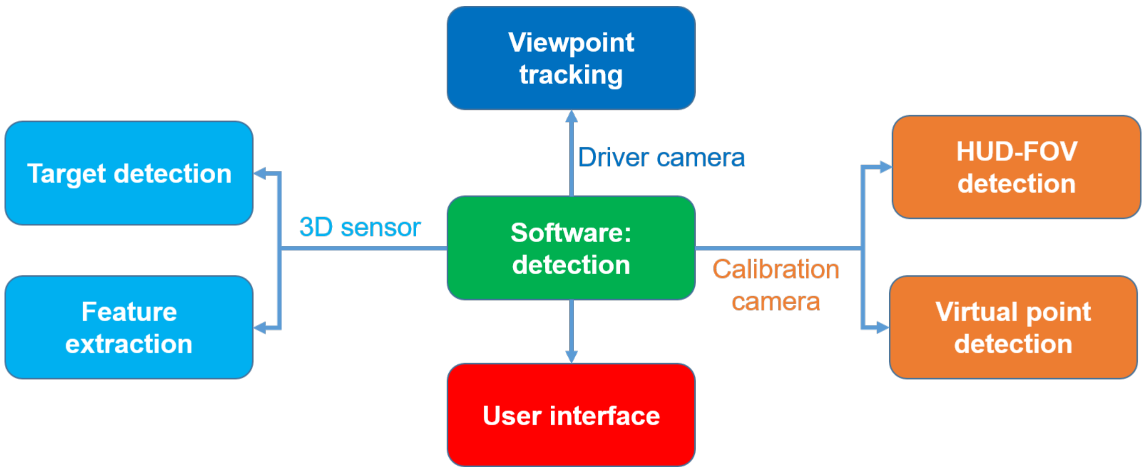
The data processing part is not directly connected to the sensors or other hardware devices. It receives data collected by the detection module and solve the unknown quantities that we want. Therefore, its tasks include:

1. solving linear perspective projection (pinhole camera) model and nonlinear distortion model using 2D–3D correspondences between the virtual image and world spaces at the selected viewpoints;
2. finish interpolation for other non-participating viewpoints;
3. acquiring the transformation from driver camera to world space using 3D–3D correspondences;
4. calculating reprojection in the validation phase and projection in the evaluation phase;
5. interacting with the operator.

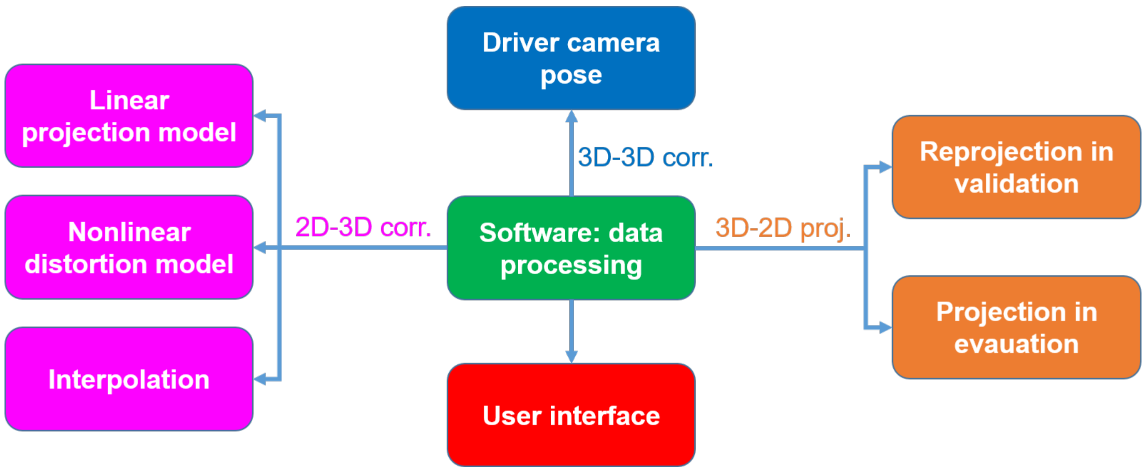
Thus, we summarize the above three software modules in the following Figure 3.8. They are developed using our home-made *Python* (Release 3.7.0) and C++ code.



(a) Controlling module.



(b) Detection module.



(c) Data processing module.

Figure 3.8: Software modules for controlling (a), detection (b) and data processing (c), respectively. corr.: correspondence(s); proj.: projection.

4. Methods

In this chapter, we explain our calibration methods in detail with their evolution. We first propose a general pipeline that outlines the essential calibration phases in Section 4.1, including the initial calibration, validation, and evaluation. The following Section 4.2 interpret our approach for driver camera calibration using the acquired extrinsic matrices. Next, we focus on the 3D sensing in Section 4.3, which is a crucial procedure in each calibration method. It involves the detection using a pre-designed target, as well as using inherent feature points from the environment. In Section 4.4, we depict the manual and automatic implementations, namely, evolution according to participation of human labor. At the end of this chapter, i.e. Section 4.5, we will demonstrate various experimental schemes under the automatic calibration concept.

4.1 General Pipeline

As is shown in Figure 4.1, an entire calibration pipeline includes three stages, i.e. initial calibration, validation, and evaluation. The detected 3D control points can be categorized into three groups, i.e. the “train”, “validation” and “test” sets. Furthermore, the detected or projected corresponding 2D virtual points are also classified into these three groups accordingly. The three terminologies frequently appear in machine learning, while we use them here for a better understanding of different procedures in the calibration.

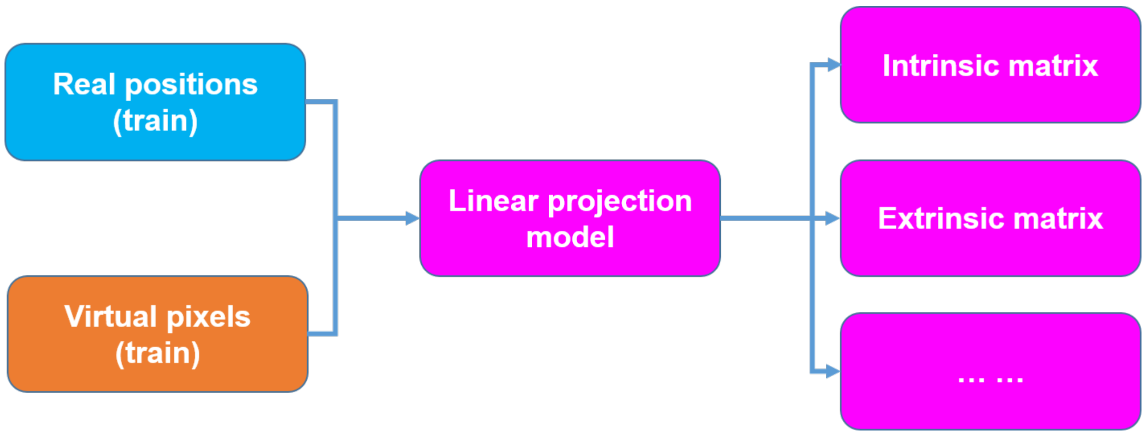
We shall finish the above three phases in the laboratory environment to examine the whole calibration concept. Comprehensive statistics are also based on such completeness and presented later in Chapter 5. However, for simplicity, we can skip the validation phase and directly enter the evaluation because the latter can be recognized as a higher level of validation. When our calibration approaches are proven effective and finally introduced to the actual automobile industry, only the initial calibration is required on the production line or in workshops, concerning the time-efficiency and workload.

Notably, this general pipeline is not rigid: some blocks in Figure 4.1 can be rearranged or replaced according to concrete calibration schemes. Such flexibility is later demonstrated in the automatic implementations in Section 4.5.

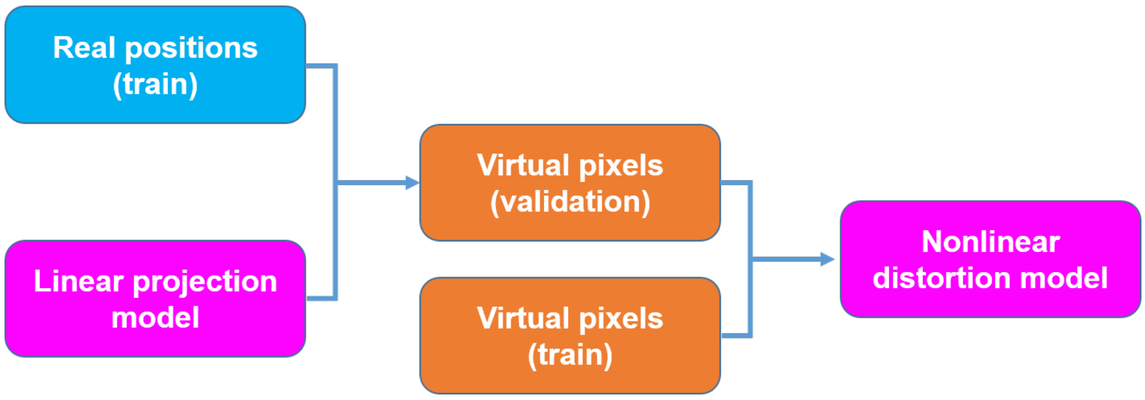
4.1.1 Pre-caliabration of Sensors

We must first pre-calibrate the connected sensors to guarantee the precision of detections before starting the AR-HUD calibration. This is particularly important for all the participating cameras because their residual optical distortion can damage our 3D reconstruction results or 2D detection based on captured pictures. Though for some employed sensors, e.g. the ZED stereo cameras, we can download their calibration files from the manufacturers' website by typing in their serial numbers[1], we still calibrate them again because of the different application environments. We pre-calibrate the following three camera systems:

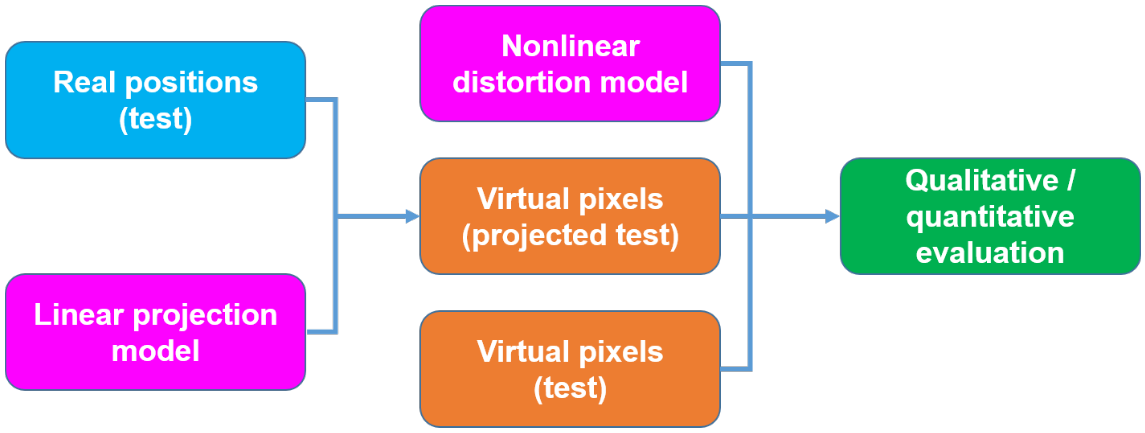
[1]`https://www.stereolabs.com/developers/calib`, Retrieved 2020-03-06.



(a) Initial calibration.



(b) Validation phase.


Real positions (test)
Linear projection model
Nonlinear distortion model
Virtual pixels (projected test)
Virtual pixels (test)
Qualitative / quantitative evaluation

(c) Evaluation phase.

Figure 4.1: An entire calibration pipeline consists of three phases: initial calibration (a), validation (b), and evaluation (c). Different colors label different data types: blue for 3D positions, orange for 2D positions, magenta for relevant parameters, and green for operations.

- the first ZED camera that serves as the 3D sensor;
- the second ZED camera that serves as the driver camera;
- the smartphone camera that serves as the calibration camera at viewpoints.

The parameters that we shall calibrate include:

- the focal lengths (f_x for horizontal and f_y for vertical) in pixel and principal points $[x_0, y_0]^T$;
- the distortion coefficients, such as those for radial (k_i) distortions;
- the transformation matrices $\mathbf{T}_{WW_R}$ from the right-eye (where we define a "right-eye" world space W_R) to the left-eye cameras (where we define the world space W as the standard) of the ZED stereo cameras, which contain the information of baselines B_{ZED}.

To unify pictures' quality, we denote that all the participating cameras run under the Full HD resolution. We apply a conventional monocular camera calibration algorithm [13] to calibrate all the single cameras, and the absolute orientation algorithm in [70] to further calibrate the transformations $\mathbf{T}_{WW_R}$. Note that this "right-world" space W_R is defined similarly to the world space W, whose origin is yet at the ZED camera's right eye. Accordingly, we print an A4 paper filled with a 10×15 chessboard pattern as the calibration target, which contains a 9×14 grid of corners. Indeed, this target is what we use later in Section 4.5.2. Alternatively, paper or films of other sizes or designs can also be exploited as long as they have enough recognizable feature points. After the pre-calibration, we acquire all the above-stated focal lengths and the baselines, and the distortion coefficients that we use to rectify captured photos in the following steps.

4.1.2 Initial Calibration

In the initial calibration, our main task is to solve the relevant unknown parameters, e.g. the intrinsic matrix $\mathbf{K}$ and extrinsic matrix $\mathbf{T}_{V_iW}$ of the pinhole camera model in Eq. (2.5)–(2.9). We first manage to estimate the intrinsic parameters, especially the focal lengths in pixel, i.e. f_u and f_v in Eq. (2.8). Then we calibrate the view-dependent extrinsic matrices by obtaining 2D–3D correspondences at the selected train viewpoints $V_{i,\text{tr}}$ in the eye box.

For a clear description, we take the operation at a single viewpoint $V_{i,\text{tr}}$ as an example. We first employ the 3D sensor to detect specific train control points $P_{j,\text{tr}}$ from a calibration target or the scene ahead of the windshield. In the practical implementation, these control points can stay invariant and be reused when the viewpoint changes. Next, we have to find their corresponding 2D points $p_{ij,\text{tr}}$ on the HUD virtual image I_i. Note that the pixel positions of these virtual points change with the viewpoint. Alternatively, we can also first display some known 2D virtual points and then reconstruct their corresponding 3D control points. Till now, we have acquired a group of 2D–3D correspondences, with which we achieve the parameters for the pinhole camera model, e.g. by using PnP algorithms. These parameters will be validated and evaluated in the next two phases.

4.1.3 Validation

Unlike the typical machine learning process, the validation does not require new control points. Instead, we can directly exploit those 3D control points $P_{j,\mathrm{tr}}$ from the train set as the validation set. In other words, we do not have to change the scene or move the target after the initial calibration. We project these control points onto the virtual image using the above-solved calibration parameters and obtain the 2D virtual points $p_{ij,\mathrm{va}}$ as the validation set. Their pixel positions shall be compared with those $p_{ij,\mathrm{tr}}$ from the train set since they are related to the same 3D control points $P_{j,\mathrm{tr}}$. Ideally, there should be no displacement from these $p_{ij,\mathrm{va}}$ to $p_{ij,\mathrm{tr}}$.

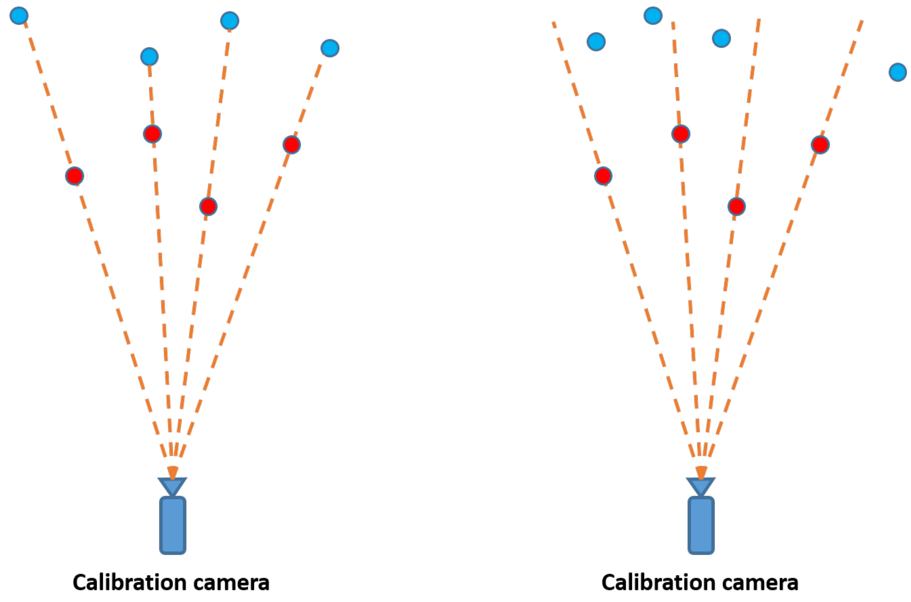
However, even if we assume no detection errors in 3D and 2D sensors, a certain amount of residual errors will appear due to the optical distortion. If the errors fall into a tolerable interval, the calibrated parameters are validated for the control points $P_{j,\mathrm{tr}}$ from the train set. Thus, another important task in the validation phase is to extract distortion parameters, which can compensate for the optical distortion in both the evaluation and the future augmented rendering. The extraction usually makes use of the above mentioned projection errors between $p_{ij,\mathrm{tr}}$ and $p_{ij,\mathrm{va}}$. In our implementations, we use the concept of warping maps that we introduced in Section 2.9.2.

Nevertheless, this validation cannot guarantee a precise augmentation across the entire HUD-FOV. There are several reasons behind this. First of all, the control points have not been updated. Put differently, only the augmentation at these given 3D positions $P_{j,\mathrm{tr}}$ are safe, whereas that at other positions may suffer a potential underfitting. Sometimes the validation demonstrates an averaged projection error of only a few pixels, but the evaluation goes far beyond a reasonable tolerance, ascribed to incompatible algorithms and target configurations. For example, when we have a planar target at a single pose in the initial calibration, using non-coplanar calibration algorithms, e.g. direct linear transformation (DLT) [37] or PnP algorithm [44], will lead to such a failure. Another tricky scenario appears when in the driver's view, new 3D positions or the test control points $P_{j,\mathrm{te}}$ are spatially collinear with those $P_{j,\mathrm{tr}}$ in the train set, as is shown in Figure 4.2 (a). In this case, the two sets fall on the same optical rays that depart from the viewpoint. The evaluation's projection errors can be tiny, but the calibration remains incomplete because the evaluation only involves the same optical rays as in the initial calibration and validation phases. At last, only the train viewpoints $V_{i,\mathrm{tr}}$ are involved till the validation phase, which is only a finite, limited part of the eye box. The further evaluation step at test viewpoints $V_{i,\mathrm{te}}$ is required for complete proof of the calibration results.

4.1.4 Evaluation

A careful evaluation is necessary for a full examination of any calibration method. Nonetheless, we are unable to evaluate the calibration parameters using all the existing real-world positions inside the HUD-FOV because there are infinitely many to sample, even for a HUD-FOV with small opening angles. Therefore, a practical compromise is to select test control points $P_{j,\mathrm{te}}$ that are non-collinear with those $P_{j,\mathrm{tr}}$ from the train set, as is sketched in Figure 4.2 (b). Hence, we can move the calibration target to another distance or pose to fulfill this condition in the implementation. Other similar measures are not exclusive.

In the evaluation phase at a train viewpoint $V_{i,\text{tr}}$, we project the test control points $P_{j,\text{te}}$ onto the virtual image I_i using the calibration parameters acquired from the initial calibration. If and only if the projected virtual points $\hat{p}_{ij,\text{te}}$ overlap the test control points within a tolerable error range, we can conclude that our calibration is precise for the given $V_{i,\text{tr}}$. Concerning multiple viewpoints inside the eye box, we should repeat the evaluation at those test viewpoints $V_{i,\text{te}}$. The corresponding extrinsic matrices and distortion models are obtained using interpolation. So far, a complete calibration of an automotive AR-HUD has been finished.



(a) Collinear case.   (b) Non-collinear case.

Figure 4.2: collinear (a) and non-collinear (b) evaluations. Red circles: selected 3D control points $P_{j,\text{tr}}$ in the initial calibration and validation phases; blue circles: control points $P_{j,\text{te}}$ in the evaluation phase. In the collinear case, the train and test control points share the same optical rays (orange) starting from the viewpoint, whereas, in the non-collinear case, they are distributed on different rays.

4.2 Multiple Viewpoints & Driver Camera Calibration

After completing the initial calibration at all the train viewpoints $V_{i,\text{tr}}$, we can further finish a driver camera calibration. The calibration is, indeed, an optional step in our pipeline. Usually, the relative pose between the driver camera space D and the world space W is pre-aligned by the vehicle manufacturer or recalibrated independently. Here, we introduce another undemanding calibration approach for the driver camera, which is integrated into the AR-HUD calibration as a by-product. From the initially calibrated extrinsic matrix $\mathbf{T}_{H_iW}$ in Eq. (2.9), we can restore the 3D positions of these train viewpoints in the world space W, denoted as $P^W_{V_{i,\mathrm{tr}}}$:

$$P^W_{V_{i,\mathrm{tr}}} = \begin{bmatrix} X^W_{V_{i,\mathrm{tr}}} \\ Y^W_{V_{i,\mathrm{tr}}} \\ Z^W_{V_{i,\mathrm{tr}}} \\ 1 \end{bmatrix} = \mathbf{T}^{-1}_{H_iW} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{H_iW} & \mathbf{t}_{H_iW} \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} . \tag{4.1}$$

Meanwhile, we also read these viewpoints through the driver camera, denoted as $P^D_{V_{i,\mathrm{tr}}}$. Then using the absolute orientation algorithm [70], we can solve the transformation $\mathbf{T}_{WD}$ from the driver camera space D to the world space W. Thus, the inherent coordinate system of the driver camera is aligned with the defined world space.

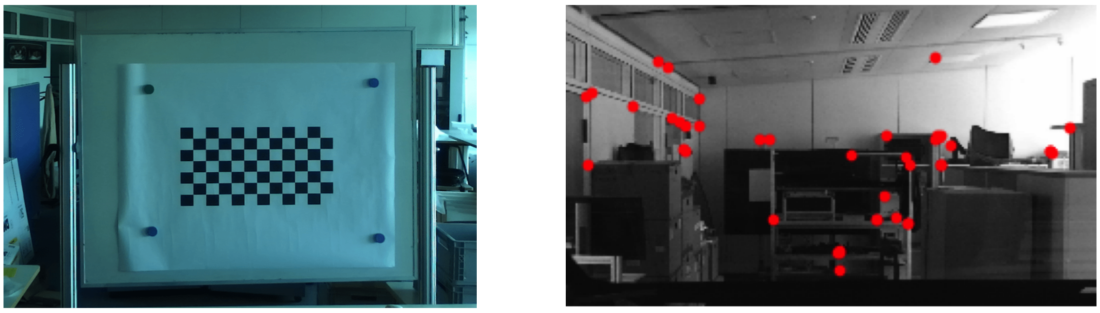
4.3 Objects for 3D Sensing

Generally, a primary task for calibration work is to select a reasonable calibration target or reference objects and then recover the positional information. In our concept, we shall make sure that enough control points are available for the 3D sensor to detect. The required minimum number of control points usually depends on the adopted calibration algorithms. For example, if we use the PnP algorithms under the pinhole model assumption, then we need at least three control points (P3P) [71]. Though in our target-free calibration method (see also Section 4.5), we employ inherent feature points from the frontal scene instead of any calibration target, the requirement on 3D sensing quality becomes even higher because we have no preknowledge of their geometry. In any case, accurate 3D sensing is an essential prerequisite for our calibration work; otherwise, there will be offsets or non-recoverable deviations in the calculated calibration parameters.

Thus, we interpret the two approaches that we will apply to fulfill the 3D detection requirement. The first option is to use a manually designed calibration target, e.g. a chessboard shown in Figure 4.3 (a). We can customize various features or markers for 3D detection, which adapt to the specific experimental environment. Furthermore, we are already aware of the geometric information while designing it, bringing useful constraints in the calculation. There have been many different kinds of such targets in the previous literature on camera calibration, such as a board with chessboard pattern [13] or concentric circles [72]. The advantage of using a specially made calibration target is that accurate 3D sensing can be realized through brilliant design. Nonetheless, it brings extra cost when the target is made of high-quality but expensive material or when it needs to be updated. This requires supports inside a factory or workshop, which also means a challenge for drivers to calibrate the AR-HUDs themselves.

As mentioned above, the second approach is to directly extract available features from the surrounding environment, i.e. the real scene in front of the windshield. Figure 4.3 (b) shows such a picture in our laboratory, on which our home-made SIFT-based program automatically discovers the feature points. This method often relies more on the accuracy of the 3D sensor and requires specific algorithms to reconstruct the spatial information.

For our case, we only need to reconstruct a static background because the calibration is offline. The most significant advantage of this approach is that customers themselves can also implement this target-free calibration routine. That means they do not have to visit a workshop or factory when they found their AR rendering biased or distorted. Indeed, this should be built upon robustly developed user interfaces, connections, and relevant application software in the vehicle. Though this approach escapes the costs of specific calibration targets, its accuracy may be influenced by many other factors because it is based on advanced image processing techniques, e.g. 3D reconstruction or feature extraction. Frequently encountered unfavorable conditions include adverse illumination, the scarcity of extractable features, and stereo matching errors.



(a) Chessboard standing in the laboratory. (b) Marked frontal scene in the laboratory.

Figure 4.3: Different objects for 3D sensing: a 7×12 chessboard pattern with 6×11 corners (a) printed on a sizable paper and a frontal scene (b) in the lab marked with extracted feature points (red circles).

4.4 Evolution of Implementation

According to the extent of the human labor that participates in acquiring 2D–3D correspondences to solve the view-dependent extrinsic matrices $\mathbf{T}_{HiW}$, we categorize our implementation into two sorts: manual and automatic calibrations. Though we cannot quantitatively estimate the proportion of the involving human labor, they differ from each other apparently: the former relies on manual alignment between virtual and control points to provide 2D–3D correspondences, while this procedure in the latter is completed automatically using related image processing techniques.

The two sorts also reflect our experimental environment's evolution - in the automatic calibration, the remaining human labor is mostly for the preparation work. We assume that the intrinsic matrix $\mathbf{K}$ in Eq. (2.8) is a preknowledge in the manual implementation; otherwise, its acquisition requires ad-hoc configurations of the calibration target or a pre-calibrated driver camera relative to the world space W. The estimation of intrinsic parameters is also detailed with automatic implementations in Section 4.5. Thus, we focus on determining $\mathbf{T}_{HiW}$ in this section.

4.4.1 Manual Implementation

The manual implementation under our concept has two sub-forms: either we use a human eye (virtual camera) to view the virtual image at the selected drivers' viewpoint or use a mounted monocular camera instead.

4.4.1.1 Using human eyes

The most straightforward manual implementation relies on the driver camera (head tracker), which recognizes the operator's face and returns the real-time position of his/her opening eye for the calibration, as is shown in Figure 4.4. Thus, the operator sits behind the windshield to observe the calibration target as well as the virtual image while keeping the pose as a driver. A group of virtual points is rendered in the virtual image. When the driver camera confirms that the eye has reached the selected viewpoint V_i under a reasonable pre-defined tolerance (e.g. a spherical domain with a radius of 5 mm), it tells the controlling software module that the operator's pose is correct. Note that we have to set such tolerance in case of head wobbling. Then the further calibration operation, such as manual shifting of virtual points, will start. The operator employs a keyboard, a gamepad, or another input device to move the virtual points to the detected control points within his/her view. Such an interactive process is similar to some manual low-complexity calibration methods for projectors [73]. In this way, we acquire the 2D–3D correspondences for solving the extrinsic matrix $\mathbf{T}_{HiW}$ in Eq. (2.9).

The main advantage is that there is no need to develop pattern recognition algorithms for the captured virtual points. This will save lots of effort in software integration and potential costs for extra sensors or markers. However, this routine is extraordinarily laborious because the operator has to maintain the head pose all the time for each selected viewpoint V_i. When the operator is relatively tall or short, even after adjusting the seat, he/she may still have to bend or extend the body to reach the viewpoints. Moreover, the operator cannot shift as many virtual points as desired, which will affect the accuracy of our calibration results. Otherwise, the entire calibration takes hours, and the operator endures too much workload and physiological discomfort. Empirically we collect only nine 2D–3D correspondences for each target distance, as detailed in Section 5.4. The accuracy is also influenced by the setting of eye-tracking tolerance, though it is usually a few millimeters. Indeed, there is an unignorable dilemma here: on the one hand, the larger the interval is, the further the real eye position may be off the assigned viewpoint, which will result in artificial errors in the final acquired calibration parameters; on the other hand, the smaller the interval is, the harder it is for the operator to keep his/her pose, which results in lots of burden for the human labor. It is recommended to use a stable head mounting device, which yet leads to extra cost.

Here, we also give an empirical estimation to show that using human eyes for the calibration is extremely time-consuming. Assuming we have:

- a 3×3 rectangular grid of viewpoints inside a 2D eye box;
- 9 virtual-control point correspondences to align at each target distance;
- 2 different distances to place the calibration target;
- at least 20 s to finish each 2D–3D alignment;

then, the complete calibration takes nearly 1 h even without the evaluation phase. For the workshop, this brings inconvenience because a worker has to calibrate several on-vehicle AR-HUDs per day, and therefore such a routine is extraordinarily uncomfortable and tedious.

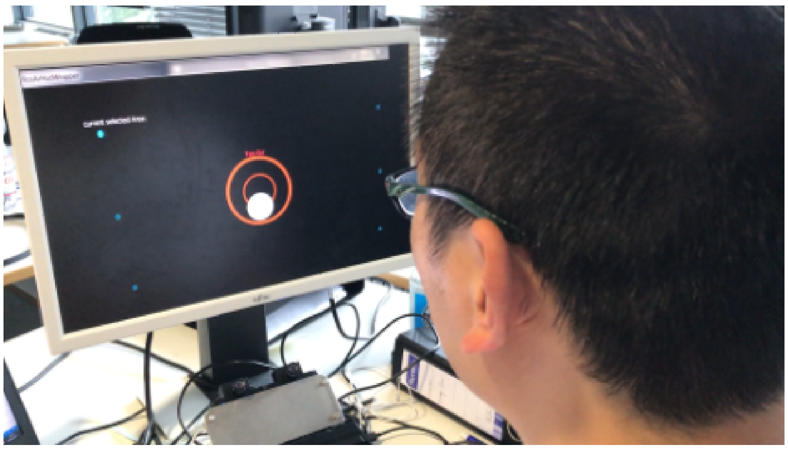


Figure 4.4: Manual AR-HUD calibration using human eyes, where a prototype driver camera is tracking the operator's eye position. For explanation with a clear picture, here, we display the real-time tracking result on a monitor. When the eye position falls to the selected viewpoint V_i within the pre-defined tolerance, the white circle will overlap the inner red ring. In actual calibration, the tracking result is shown to the operator directly via the HUD. When the eye position is confirmed correct, the operator can start shifting the virtual points by hand.

4.4.1.2 Using a Camera

Another version of manual implementation is realized by replacing the human eye with a monocular calibration camera, e.g. a smartphone camera or a webcam. As is shown in Figure 4.5, the camera is placed on a mounting setup. It is connected to the laptop or a monitor with a cable or wirelessly so that the operator can watch the real-time streaming video while sitting beside the experimental setup. The calibration camera is labeled so that our driver camera can trace its position under adapted programming. Compared to the implementation using human eyes, the fear of losing the correct view is effectively circumvented because the mounted camera stays stable at the selected viewpoint V_i. Thus, it leads to less time-consumption, allowing the operator to employ more 2D–3D point correspondences. For example, we can design a calibration target with a 4×7 grid of control points and use the same number of virtual points.

4.4.2 Automatic Implementation

In the manual implementations, the shifting of 2D virtual points to align the 3D control points is accomplished by human labor, even though we can use a calibration camera to replace human eyes. Since the operator has to move the virtual points one by one, the manual implementations restrict both the time efficiency and the amount of applicable 2D–3D point correspondences. In workshops, we also have to be careful of fatigue resulting in artificial errors and finally affecting the calibration accuracy. Besides, it is hard to extract

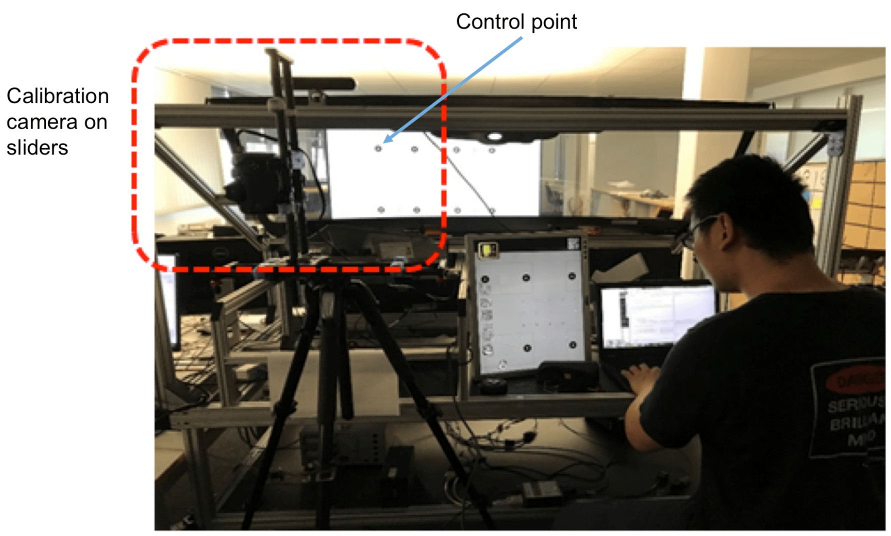


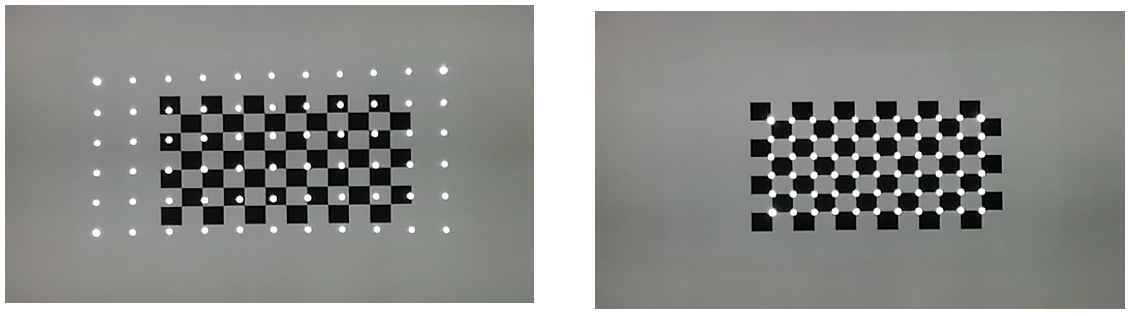
Figure 4.5: Manual AR-HUD calibration using a camera. In this instance, we use a tripod and sliders to mount the calibration camera at a selected viewpoint V_i and connect it to a monitor. The operator can shift the virtual points much more comfortably and efficiently than directly using his/her eyes.

a precise distortion model using too few 2D–3D point correspondences. At a later stage, we will investigate how the number of 2D–3D correspondences influences the calibration's performance (see Section 5.2.4).

In contrast, in the automatic implementation, we assign the task of moving virtual points entirely to the controlling and detection software modules (Figure 3.8 (a–b)). In other words, the operator does not have to care about the manual shifting of the virtual points anymore. Our software can analyze the real-time video stream or captured pictures from the calibration camera, and then automatically finish and confirm the alignment between 2D virtual points and 3D control points all at once. For this purpose, we employ some image processing techniques, such as contour detection to recognize virtual points' centroids. We also intensively call some relevant functions from the *OpenCV* library [44]. After a few iterative rounds, the virtual points will automatically superimpose the detected 3D control points, as is shown in Figure 4.6. This approach is adopted later in our automatic calibration scheme using a large chessboard target, as is detailed in Section 4.5.1.

Alternatively, we can first display a grid of virtual points with known pixel positions p_{ij} in the virtual image I_i, and then reconstruct their corresponding control points P_{ij} by calculating the connecting optical rays $\mathbf{r}_{ij}$. We realize this thought later with the calibration using a piece of patterned paper in Section 4.5.2 and the target-free calibration in Section 4.5.3, respectively.

Compared to the manual implementation, the automatic approaches save much time in running the experiment. As is stated before, it usually takes 20 s to finish aligning a single virtual point manually when we calibrate our AR-HUD using human eyes, but the automatic point shifting can align over a hundred virtual points within a few seconds.



(a) A 7 × 12 chessboard (6 × 11 corners) with an initially rendered 6 × 11 virtual point array.

(b) After automatic shifting of the virtual points, all of them overlap the chessboard corners.

Figure 4.6: Automatic AR-HUD calibration using a chessboard. Here the chessboard corners serve as the control points while the HUD renders the virtual points. After automatic aligning, these virtual points are located precisely at the control points.

That is why we can deploy many more 2D–3D point correspondences in the HUD-FOV. The more such correspondences we have, the more accurate calibration parameters and distortion models we expect to restore.

Nevertheless, the automatic calibration schemes are still not a perfect solution. The performance of image processing techniques depends on the local environment, especially the illumination condition. If the lighting is too bright, it is hard for our algorithm to recognize the virtual points; otherwise, the detection of calibration target or natural feature points can be unstable if it is too dim. Hence, one should be careful of the illumination condition in the preparation work. For different HUD types, some critical parameters in the algorithm shall also be re-adapted, e.g. the brightness filter threshold. Moreover, not each 2D–3D alignment is collected as accurately as that under the manual implementation because the automatically recognized centroids of the virtual points may deviate the corresponding control point for a few pixels. Such a 2D recognition error is investigated later in Section 5.2.3.

Table 4.1 compares our manual and automatic AR-HUD calibration implementations empirically and qualitatively w.r.t. various aspects.

4.5 Experimental Schemes

In this section, we describe three examples of our latest developed automatic calibration schemes in detail. The first is to use a relatively sizeable patterned board as the target, which is placed in front of the windshield at several meters. The second one is to apply a small sheet of patterned paper as the calibration target, which is laid onto the windshield's outer surface. The last example is a target-free implementation, where we rely no more on any pre-designed calibration target but extract the inherent graphical features from the frontal scene for 3D sensing.

All of them are finished using a smartphone as the calibration camera at viewpoints. In principle, the manual routine can also be conducted under similar schemes, regardless

Table 4.1: Comparison between manual and automatic implementations in several aspects. Note that this comparison is qualitatively and based on our experience in the laboratory environment.

Sort of implementation	**Manual**	**Automatic**
Image receiver	Human eye or camera	Camera
Sort of target	Pre-designed target	Pre-designed target or target-free
Intrinsic parameters	Need preknowledge	Estimated using pictures
Acquisition of 2D–3D correspondences	Manually aligned	Automatic aligned or reconstructed
Number of 2D–3D correspondences	< 30	> 100
Particular error sources	Tolerance of eye position	Environmental factors, e.g. illumination
Distortion correction	Difficult	Warping maps applicable
Accuracy of results	Restricted by number of correspondences	Enhanced by larger number of correspondences
Time efficiency	Low	High

of human workload. Note that we have completed all these schemes in a laboratory environment. Nonetheless, we will later show their effectiveness with experimental results in Chapter 5. We also have published these three methods as [24], [56], [57], respectively.

4.5.1 Calibration with a Chessboard Target

We first present a calibration method for AR-HUDs using a sizable chessboard and warping maps. The intrinsic matrix of HUD is estimated using a constructed stereo vision. By automatically shifting 2D virtual points to 3D chessboard corners within the smartphone camera's view, we obtain 2D–3D correspondences and then compute view-dependent extrinsic parameters. Using the obtained parameters, we reproject the chessboard corners to the virtual image. Comparing the reprojected with ground truth virtual points, we acquire 2D bias vectors, from which we reconstruct a series of warping maps for compensating optical distortions. We further obtain the corresponding extrinsic parameters and warping maps through interpolation for any other uninvolved viewpoint in the eye box. Besides, we calibrate the driver camera by utilizing the acquired extrinsic parameters and viewpoint tracking results.

4.5.1.1 Preparation

In this scheme, the chessboard pattern is placed in front of the windshield at different distances to the viewpoints. It is shown on a 140 cm $\times$ 80 cm large screen (Samsung UE65MU6199UXZG) with proper mounting devices. Generally, any other featured pattern can also serve the same role. For simplicity and lower cost, we can also use a giant paper chessboard instead, but using the screen is flexible to change the chessboard geometry.

We adopt the conventional non-coplanar camera calibration algorithm to solve the unknown parameters. This requires that not all the control points fall on/near a common plane in the space. For this purpose, we set upright the chessboard to two sufficiently different distances (approximately 3.0 m and 3.8 m to the ZED stereo camera) so that the control points are distributed on different planes, as is sketched in Figure 4.7. More than two distances are also applicable. The implementation consists of several steps in sequence: estimation of the intrinsic matrix $\mathbf{K}$, calibration of extrinsic matrices $\mathbf{T}_{HiW}$ at the train viewpoints $V_{i,\mathrm{tr}}$, calibration of driver camera, restoration of distortion characters, and finally, interpolations for the test viewpoints $V_{i,\mathrm{te}}$.

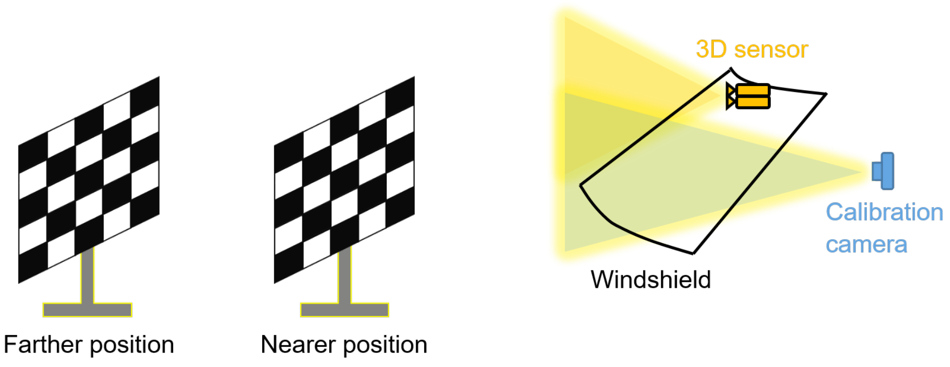


Figure 4.7: Configuration of two sufficiently separated planar chessboard targets, whose corners form a group of non-coplanar 3D control points P_j. The two half-transparent triangles illustrate the FOVs of 3D sensor and calibration camera, respectively.

We define the origin of world space W at the optical center of the 3D sensor's left eye. At each distance, the chessboard is approximately parallel to the $Y^W Z^W$–plane. We adjust the chessboard design and its position at the nearer distance so that the control points are roughly homogeneously distributed in the HUD-FOV. This is useful in obtaining an unbiased warping map at a later stage. Although this tricky condition should also be fulfilled at the farther distance, the screen size cannot fill out the HUD-FOV in this case.

The eye box is selected as an area in the $Y^W Z^W$–plane, plotted in Figure 4.8. As is stated in Section 3.2, the calibration is insensitive to the X^W–component of viewpoints when the HUD's opening angles are small. Inside the eye box, we select a grid of 18 viewpoints as the train set, with 20 mm between neighboring ones. We also select 10 test viewpoints, at which we will evaluate our interpolation results. Usually, placing the chessboard and mounting the smartphone camera takes less than 10 min. Before the next steps, all the involving cameras, including the smartphone and stereo cameras, should be pre-calibrated.

4.5.1.2 Estimation of Intrinsics

We estimate the intrinsic matrix $\mathbf{K}$ using a stereo method with the chessboard target. As is shown in Figure 4.9 (a), by controlling the moving stage, we place the smartphone camera sequentially at two viewpoints that are horizontally separated by a baseline distance of 120 mm. We take two pictures of a chessboard pattern displayed on the screen,

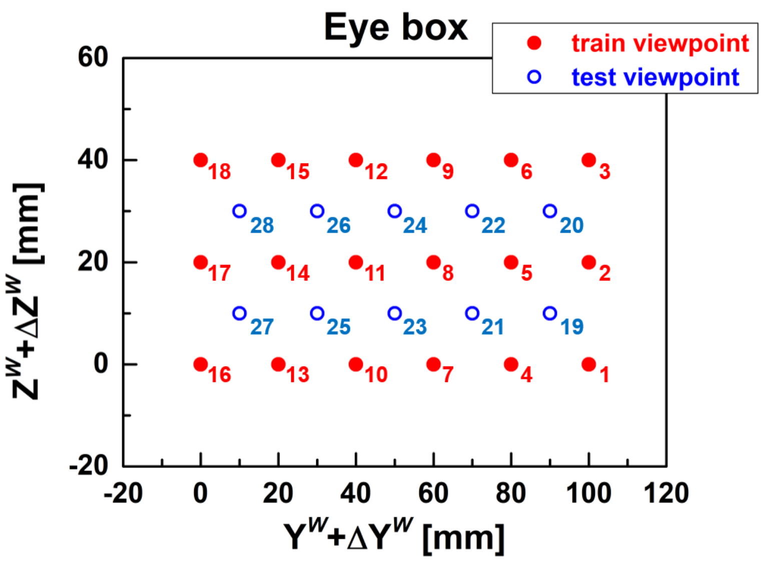
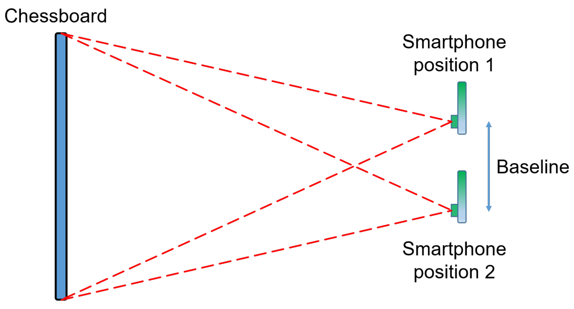
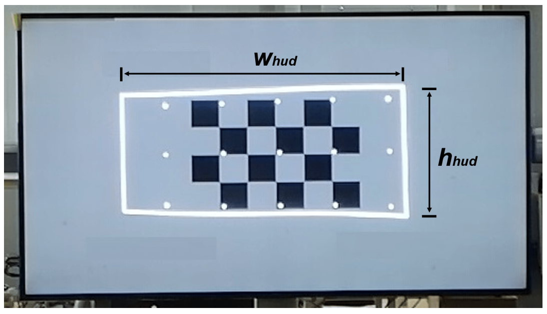


Figure 4.8: The selected 2D eye box with 18 train (red circle) and 10 test (blue ring) viewpoints on the $Y^W Z^W$–plane. ΔY^W and ΔZ^W represent the offsets on the corresponding two axes.

respectively. Then we can acquire the depth d_{board} using triangulation from the stereo vision. Meanwhile, since we have known the geometry of the chessboard and detected the frame of virtual image (i.e. white frame in Figure 4.9 (b)), we can calculate the physical width w_{hud} and height h_{hud} of the HUD-FOV on the chessboard plane. In the end, we estimate the horizontal (f_u) and vertical (f_v) focal lengths of our HUD using the following equations:



(a) Constructed stereo vision to measure the chessboard.



(b) Frame of virtual image (HUD-FOV's boundary) on the chessboard plane.

Figure 4.9: Estimation HUD's focal lengths by constructing a stereo vision (a) to capture a displayed chessboard pattern (b).

$$f_u = d_{\text{board}} \cdot \frac{u_{\max}}{w_{\text{hud}}} \ , \quad f_v = d_{\text{board}} \cdot \frac{v_{\max}}{h_{\text{hud}}} \ , \tag{4.2}$$

where $u_{\max} \times v_{\max}$ denotes the virtual image's resolution.

We assume that the principal point $[u_0, v_0]^T$ falls close to the image center, and the HUD projector's pixels are rectangular, as is in line with conventional designs of automotive HUDs. That means in the intrinsic matrix $\mathbf{K}$ in Eq. 2.8, we have $s = 0$

and $[u_0, v_0]^T = [u_{\max}/2, v_{\max}/2]^T$. These approximations will be justified later with our experimental results in Section 5.5.1. In general, other possible values are not excluded, depending on the concrete design of the HUD optics or practical measurements.

4.5.1.3 Calibration of Extrinsics

We have two rounds for the two target positions, for which we utilize a 7×12 chessboard pattern with 6×11 corners as the control points. In each round, we detect the corners' 3D positions in the world space W and reuse them for every train viewpoint $V_{i,\mathrm{tr}}$. The respectively aligning 2D virtual points $p_{ij,\mathrm{tr}}$ are obtained as follows: we render a 6×11 grid of virtual points via the HUD and shift them automatically using the software that is also connected to our smartphone camera. Finally, in the camera image, all the virtual points overlap with the control points, as is shown above in Figure 4.6. After the two rounds, we have 132 2D–3D correspondences at the two target distances for every $V_{i,\mathrm{tr}}$. Thus, the calibration of the extrinsic matrix $\mathbf{T}_{H_iW}$ becomes a non-coplanar PnP problem, which we solve using functions in the *OpenCV* library [44]. Then, we perform a driver camera calibration according to the method in Section 4.2.

4.5.1.4 Validation & Warping Maps

As is stated in Section 2.9.2, for each train viewpoint V_i, we compute its corresponding warping maps $w_{i,\Delta u}$ and $w_{i,\Delta v}$ from the validation phase. Since we have obtained the intrinsic $\mathbf{K}$ and extrinsic matrix $\mathbf{T}_{H_iW}$, we project the 3D control points P_j onto the virtual image I_i to acquire the reprojected virtual points $p_{ij,\mathrm{va}}$. Then we compute the bias vectors using the following equation analogous to Eq. (2.40):

$$\mathbf{v}_{ij,\mathrm{bias}} = [\Delta u_{ij,\mathrm{rep}}, \Delta v_{ij,\mathrm{rep}}]^T = [u_{ij,\mathrm{tr}} - u_{ij,\mathrm{va}}, v_{ij,\mathrm{tr}} - v_{ij,\mathrm{va}}]^T . \tag{4.3}$$

Indeed, both groups of virtual points, i.e. $p_{ij,\mathrm{tr}}$ and $p_{ij,\mathrm{va}}$, are already affected by the optical distortion. Since the former is obtained from detection while the latter from calculated reprojection, we regard $p_{ij,\mathrm{tr}}$ as ground truth. As we have assumed, the optical distortions should vary smoothly within the virtual image I_i. Therefore, the warping map $w_{i,\Delta u}$ and $w_{i,\Delta v}$ are differentiable at each inner pixel. Consequently, we reconstruct the whole warping maps using bicubic interpolation with those bias vectors.

4.5.1.5 Interpolation among Viewpoints

An ideal AR-HUD calibration routine should be valid at not only the train viewpoints $V_{i,\mathrm{tr}}$, but also other non-participating ones inside the eye box. However, it is impractical to repeat calibration measurements at all these countless viewpoints. Out of this concern, we introduce our interpolation concept among viewpoints as a solution, which includes two aspects: one is the interpolation for extrinsic matrix $\mathbf{T}_{H_iW}$, while the other is the interpolation of warping maps $w_{i,\Delta u}$ and $w_{i,\Delta v}$.

The raw interpolation of extrinsic matrices is accomplished as follows: after we finish calibration for all the 18 train viewpoints, we use their corresponding $\mathbf{R}_{H_iW}$ and $\mathbf{t}_{H_iW}$

(see also Eq. 2.9) to interpolate the extrinsic parameters at a non-participating viewpoint through bilinear interpolation. A convenient way to interpolate $\mathbf{R}_{H_iW}$ is first to convert them to the rotation vectors $\mathbf{r}_{H_iW}$, interpolate these vectors, and then convert the results back to matrices.

We further obtain warping maps for non-participating viewpoints. Assuming that warping maps change smoothly within the eye box, for any non-participating viewpoint, we compute its warping maps through bilinear interpolation using those $w_{i,\Delta u}$ and $w_{i,\Delta v}$ at the train viewpoints. Later for distortion correction in AR rendering, we can extract the corresponding bias vector $\mathbf{v}_{ij,\text{bias}}$ from the warping maps and dewarp the projected virtual image, as is interpreted in Section 2.9.2.

We apply these interpolation approaches later to evaluation steps at the test 10 viewpoints $V_{i,\text{te}}$. Those relevant results are shown in Section 5.5.1.4.

4.5.1.6 Summary

The above AR-HUD calibration scheme employs a conventional sizable chessboard as the target. It applies viewpoint-based interpolation and warping-map-based distortion correction. These new concepts in AR-HUD calibration circumvent extensive sampling on hundreds of viewpoints in the polynomial-based calibration models [10], [18]. Meanwhile, using the chessboard pattern and the smartphone, this scheme depends less on particular components such as on-vehicle markers or textured cover in [10], or HoloLens glasses in [18].

It is worthy to note that in our method, the chessboard should be designed appropriately. Its corners should cover the HUD-FOV as densely and homogeneously as possible so that the bicubic interpolation to generate warping maps is accurate enough, as is stated in Section 2.8. Even then, setting a proper target pattern for the full HUD-FOV requires a flexible yet stable mounting, which may result in extra labor and costs. Indeed, the chessboard's dimension is also dependent on the distances where it stands. For a fixed distance, the larger the HUD-FOV is, the larger the chessboard should be; for a fixed HUD-FOV, the larger the distances are, the larger the chessboard should be.

4.5.2 Calibration with Patterned Paper

To avoid using a sizable calibration target, we propose here a low-complexity yet accurate calibration scheme using only a small sheet of printed patterned paper as the target, which is laid directly on the windscreen. The full HUD-FOV can be calibrated, with the optical distortion corrected by warping maps. The issue of changing views is again resolved by interpolating both projection parameters and warping maps.

Particularly, we utilize the decomposition of the extrinsic matrix $\mathbf{T}_{H_iW}$ in Eq. (2.7):

$$\mathbf{T}_{H_iW} = \mathbf{R}_{H_iV}\mathbf{T}_{V_iW}, \tag{4.4}$$

indicating that the transformation from the world space W to the current HUD-FOV space H_i consisting of:

- the transformation from W to the current viewpoint space V_i;
- the rotation from V_i to the current H_i.

Our inspiration is to calibrate the above-stated three matrices individually. Indeed, the calibration of $\mathbf{T}_{V_iW}$ can be somewhat recognized as an independent routine because it is purely solving the calibration camera's relative pose and not related to the HUD-FOV or virtual images. The HUD can keep switched off in this step.

In terms of implementation, when we lay the patterned paper on the windshield, it can be seen by the smartphone camera, yet not by the 3D sensor. However, when it is off the windshield, we can place it inside the overlapping FOV between the 3D sensor and the smartphone camera so that both the sensors see it. Therefore, when the paper stays onside the windshield, we can switch on the HUD and calibrate the rotation matrix $\mathbf{R}_{H_iV_i}$; otherwise, we can use the paper target to calibrate the transformation $\mathbf{T}_{V_iW}$.

4.5.2.1 Preparation

We print an A4 paper filled with a 10×15 chessboard pattern as the calibration target, which contains a 9×14 grid of corners, as is shown in Figure 4.10. Of course, paper or films of other sizes or designs can also be used as long as they are large enough to cover the HUD-FOV and have an adequate number of detectable feature points. Next, we choose a 5×5 grid that covers an $80\,\text{mm} \times 60\,\text{mm}$ area as our eye box, identical to the example in Figure 3.2 (a). We select 9 viewpoints as the train and the other 16 as the test set. For any viewpoint V_i inside, the entire HUD-FOV remains unblocked in the captured video.

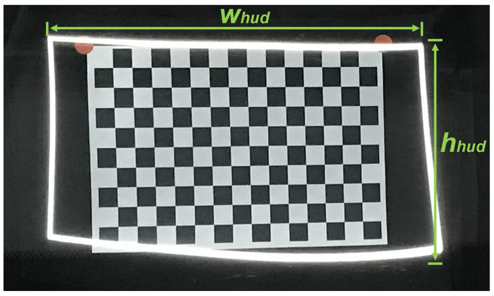
The origin of world space W is still defined at the optical center of the 3D sensor's left eye, like in Section 4.5.1.1. Before the next steps, all the involving cameras, including the smartphone and stereo cameras, are pre-calibrated.

4.5.2.2 Estimation of Intrinsics

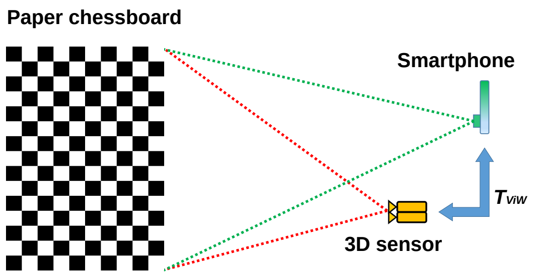
To estimate the intrinsic matrix $\mathbf{K}$ of our HUD, we still adopt the assumption in Section 4.5.1, i.e. the skew factor is zero and the principal point $[u_0, v_0]^T$ falls at the center of virtual image. Then, the remaining parameters to estimate are the horizontal and vertical focal lengths in pixel, i.e. f_u, and f_v. Here we hold upright the chessboard paper in the HUD-FOV and adopt the estimation approach in Section 4.5.1.2. We display the HUD-FOV borders (Figure 4.10 (a)) and then take photos at two viewpoints on the Y^W–axis yet separated by a baseline distance. In this case, we have formed a stereo vision and can obtain the depth d_{board} of the chessboard by detecting its corners. Using the chessboard geometry, we estimate the physical width w_{hud} and height h_{hud} of HUD-FOV at the distance d_{board}, and further acquire the focal lengths using Eq. (4.2).

4.5.2.3 Transformation from World to Viewpoint

The next step is to determine the transformation $\mathbf{T}_{V_iW}$ from the world W to the train viewpoint $V_{i,\text{tr}}$. As is shown in Figure 4.10 (b), we hold the same chessboard paper in the joint FOV of the smartphone and stereo camera. By actuating the moving stage,



(a) Hand-held chessboard inside HUD-FOV.



(b) Chessboard detection in multiple views.

Figure 4.10: Estimation of HUD focal lengths (a) and determination of transformation from the world to one of the viewpoint coordinate systems (b).

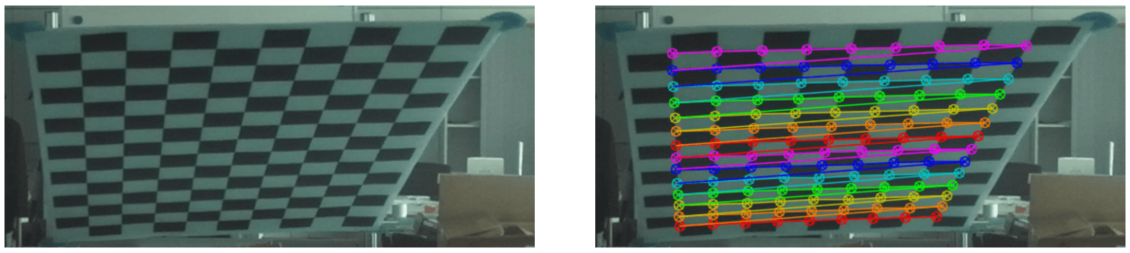
we capture the chessboard photos at all train viewpoints and find its corners. Using the disparities and triangulations, we calculate those corner positions in each viewpoint frame $V_{i,\mathrm{tr}}$. Meanwhile, we triangulate those corners in the world W using the ZED stereo camera photos. In such a way, we obtain a group of 3D–3D correspondences between W and each $V_{i,\mathrm{tr}}$, and further determine those transformations using the absolute orientation algorithm [70].

4.5.2.4 Rotation from Viewpoint to HUD-FOV

In this step, we utilize the concept of reflection points in Figure 2.1 to calibrate the rotation matrices $\mathbf{R}_{H_iV_i}$ at the train viewpoints $V_{i,\mathrm{tr}}$. Since the intrinsic matrix $\mathbf{K}$ is already estimated, we convert this sub-task into a PnP problem. To this end, we need to find a group of 2D–3D correspondences between the virtual image I_i and viewpoint space $V_{i,\mathrm{tr}}$. The 2D points are chosen as a configured grid of virtual points $p_{ij,\mathrm{tr}}$ whose pixel positions are known as input for the PGU, while the 3D ones are their corresponding restored reflection points $Pr_{ij,\mathrm{tr}}$.

Next we reconstruct these reflection points for each train viewpoint. For this purpose, we apply an indirect method by firstly calculating the shape of 3D windshield surface $S_{i,\mathrm{tr}}$ where $Pr_{ij,\mathrm{tr}}$ are distributed. We stick the chessboard paper to the windshield's outer surface at the four paper corners, as shown in Figure 4.11 (a). The inner surface is not readily applicable because the smartphone camera cannot observe virtual points clearly. From captured photos at multiple viewpoints, we can find the chessboard corners (Figure 4.11 (b)) and reconstruct their 3D positions in $V_{i,\mathrm{tr}}$. With these data, we obtain a fitted quadratic equation that describes the windshield outer surface $S_{i,\mathrm{tr}}$ in the space $V_{i,\mathrm{tr}}$, including the area which intersects the HUD-FOV. After that, we display the virtual points $p_{ij,\mathrm{tr}}$ (e.g. the 7×5 grid in Figure 4.12 (a)). Using adaptive grayscale thresholding and contour detection, we locate their centroids in the smartphone photos (Figure 4.12 (b)). These centroids are regarded as the captured projection of reflection points $Pr_{ij,\mathrm{tr}}$.

As we have already pre-calibrated the smartphone camera, we know about its intrinsic matrix $\mathbf{K}_C$. Therefore, we can calculate the parameterized optical ray $\mathbf{r}_{ij,\mathrm{tr}}$ for each



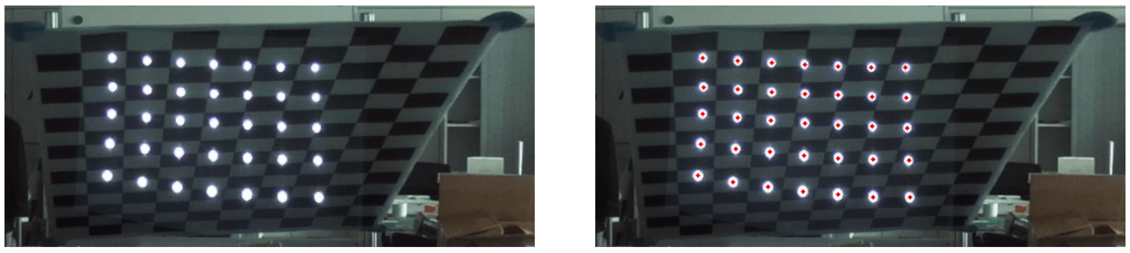
(a) Chessboard paper sticked on the outer surface of windshield (b) Located chessboard corners, which are marked and linked by rainbow lines

Figure 4.11: A printed chessboard (a) is laid on the windshield and captured by the smartphone camera. Corners are detected (b) for further reconstruction of the windshield surface.

centroid $c_{ij,\text{tr}} = [x_{ij,\text{tr}}, y_{ij,\text{tr}}, 1]^T$ (homogeneous form), which starts from the origin of $V_{i,\text{tr}}$ and points to the reflection point $Pr_{ij,\text{tr}}$:

$$\mathbf{r}_{ij,\text{tr}} = w \begin{bmatrix} X^{V_i,\text{tr}}/w \\ Y^{V_i,\text{tr}}/w \\ 1 \end{bmatrix} = \mathbf{K}_C^{-1} c_{ij,\text{tr}} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{ij,\text{tr}} \\ y_{ij,\text{tr}} \\ 1 \end{bmatrix}, \tag{4.5}$$

where f_x and f_y indicate the camera's focal lengths in pixel, $[x_0, y_0]^T$ is the principal point, and w is a scaling factor. Usually, the thickness of a windshield is up to a few millimeters, which is three to four orders of magnitude smaller than the HUD's focal length that is up to meters. Hence, we estimate the reflection points $Pr_{ij,\text{tr}}$ as the intersection between the outer surface $S_{i,\text{tr}}$ and the ray $\mathbf{r}_{ij,\text{tr}}$. These approximated $Pr_{ij,\text{tr}}$ and 2D virtual points $p_{ij,\text{tr}}$ form a group of 2D–3D correspondences, with which we can further calculate $\mathbf{R}_{H_iV_i}$ using relevant algorithms, such as *solvePnP* function in the *OpenCV* library [44] and then Levenberg–Marquardt optimization method. Now according to Eq. (2.6)–(2.7), we can finally obtain the projection matrices $\mathbf{P}_{I_iV_i}$ and $\mathbf{P}_{I_iW}$ for the train viewpoints $V_{i,\text{tr}}$.



(a) A 7×5 virtual point grid. (b) Centroids of virtual points.

Figure 4.12: An array of configured virtual points with the chessboard paper as the background is captured in a smartphone photo (a). Their centroids (b) are localized using image processing techniques.

4.5.2.5 Warping Maps & Interpolation

In Figure 4.10 (a), we can observe optical distortion because the white rectangular borders of the virtual area are askew and bent. To mitigate this artifact, we extract warping maps through reprojection by applying the concept in Section 2.9.2. However, in this step, the transformation $\mathbf{T}_{V_iW}$ becomes irrelevant because its calibration did not involve any 2D virtual pixel in I_i. Instead, we use directly those reconstructed 3D reflection points $Pr_{ij,\mathrm{tr}}$ from $V_{i,\mathrm{tr}}$. Feeding the projection matrix $\mathbf{P}_{I_iV_i}$ into Eq. (2.6), we project all $Pr_{ij,\mathrm{tr}}$ onto the virtual image I_i. Then we acquire bias vectors $\mathbf{v}_{\mathrm{bias}} = [\Delta u_{ij,\mathrm{rep}}, \Delta v_{ij,\mathrm{rep}}]^T$ pointing from the reprojected virtual points $p_{ij,\mathrm{va}}$ to the ground truth input $p_{ij,\mathrm{tr}}$. We still reconstruct the corresponding warping maps $w_{i,\Delta u}$ and $w_{i,\Delta v}$ using the approach in Section 4.5.1.5. As we have assumed in Section 2.9.2, these warping maps change continuously within the domain of eye box. Therefore the maps at uninvolved viewpoints should be obtainable through interpolation that is similar to in Section 4.5.1: for any new viewpoint, we can first interpolate $\mathbf{T}_{V_iW}$ and $\mathbf{R}_{H_iV_i}$, and then the warping maps. Particularly, the interpolation of rotational parts are done via conversion to Rodrigues rotation vectors.

4.5.2.6 Summary

The flowcharts in Figure 4.13 outline this calibration approach using readily available patterned paper. The workflows accomplish step by step the acquisition of linear matrices for perspective projection and the reconstruction of nonlinear warping maps for distortion correction. Due to the reduced effort and simplified equipment, our method opens the way for customers to recalibrate their AR-HUDs themselves. To this end, we can further develop supporting user interfaces in the vehicle.

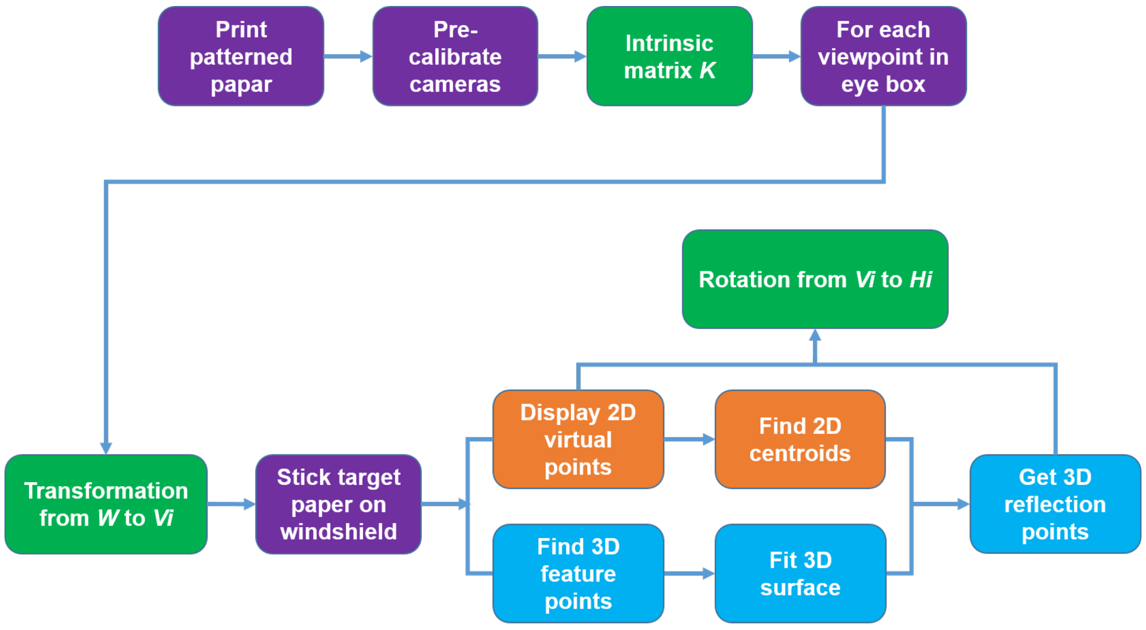
Although this scheme is low-complexity, there is a risk of losing accuracy because the recovery of transformation $\mathbf{T}_{V_iW}$ relies much on the 3D sensor. Since the distortion correction and the determination of the above transformation are decoupled during implementation, the extracted warping maps cannot compensate for any potential deviation in $\mathbf{T}_{V_iW}$. Therefore in this scheme, the requirement on the 3D sensor's quality becomes higher than in Section 4.5.1; otherwise, the virtual objects can be inaccurately located in the final AR rendering.

4.5.3 Target-free calibration

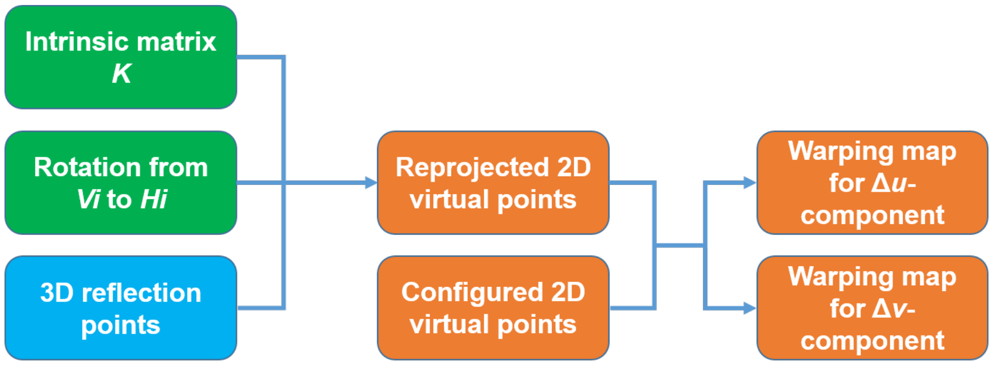
This scheme presents a target-free calibration method for AR-HUDs, which is also finished automatically using the smartphone camera. Regarding target complexity, the target-free fashion is more advanced than the above two that still rely on calibration targets. The same as in Section 4.5.2, we decouple the perspective projection matrix $\mathbf{P}_{I_iW}$ into three parts: intrinsic matrix $\mathbf{K}$, relative pose $\mathbf{T}_{V_iW}$ between the 3D sensor and the smartphone camera, and then rotation $\mathbf{R}_{H_iV_i}$ between the viewpoint and the HUD-FOV spaces. We directly exploit feature points from the real scene inside the joint FOV of the 3D sensor and the smartphone camera to obtain the relative pose. The other two threads, i.e. the determinations of intrinsic and rotational parts, are also accomplished without any ad-hoc target. They are even irrespective of environmental feature points.

4.5.3.1 Preparation

In this step, we do not have to prepare any designed target. Therefore, we focus on setting up the moving stage and the selection of an appropriate eye box. The deployed eye box


Print patterned papar
Pre-calibrate cameras
Intrinsic matrix *K*
For each viewpoint in eye box
Rotation from *Vi* to *Hi*
Display 2D virtual points
Find 2D centroids
Transformation from *W* to *Vi*
Stick target paper on windshield
Get 3D reflection points
Find 3D feature points
Fit 3D surface

(a) Procedures from start till acquisition of linear matrices.


Intrinsic matrix *K*
Rotation from *Vi* to *Hi*
3D reflection points
Reprojected 2D virtual points
Configured 2D virtual points
Warping map for Δ*u*-component
Warping map for Δ*v*-component

(b) Reprojection and reconstruction of warping maps.

Figure 4.13: Flowcharts to summarize our calibration scheme using a piece of patterned paper, including the acquisitions of linear matrices (a) and warping maps (b). Colors in blocks indicate different phases. Purple: operation or sub-tasks; green: linear matrices; orange: data or operation in 2D spaces; light blue: data or operation in 3D spaces.

and moving stage are identical to those in Figure 3.2 (a) and (b), respectively. We still divide the viewpoints into 9 train and 16 test ones.

In the laboratory, we find a proper frontal scene that contains some geometrical features, as is the captured background in Figure 4.14. Those corners of shelves, ceiling pieces and other objects are potential feature points to localize. Otherwise, it is not simple to extract useful feature points from a rather texture-free frontal scene.

4.5.3.2 Estimation of Intrinsics

We provide here a different method to estimate the intrinsic matrix than in Section 4.5.1.2 and 4.5.2.2, which does not rely on stereo vision. It is also a target-free estimation method. As is shown in Figure 4.14, the margins of virtual image area is captured by the smartphone camera. Inside the photo, we can search for the 4 corners of the white frame, which are denoted as $[x_1, y_1]^T$ (upper left), $[x_2, y_2]^T$ (upper right), $[x_3, y_3]^T$ (bottom left) and $[x_4, y_4]^T$ (bottom right). Since we also know the resolution $u_{\max} \times v_{\max}$ of the virtual image and the smartphone camera's focal lengths (f_x and f_y), then the horizontal (f_u) and vertical (f_v) focal lengths are approximately:

$$\begin{cases} f_u \approx u_{\max} \cdot \dfrac{2f_x}{x_2 - x_1 + x_4 - x_3} \\ f_v \approx v_{\max} \cdot \dfrac{2f_y}{y_3 - y_1 + y_4 - y_2} \end{cases} . \tag{4.6}$$

Then we assume again that the principal point $[u_0, v_0]^T$ in $\mathbf{K}$ lies at the center of virtual image, i.e. $[u_{\max}/2, v_{\max}/2]^T$, and the skew factor s is zero.

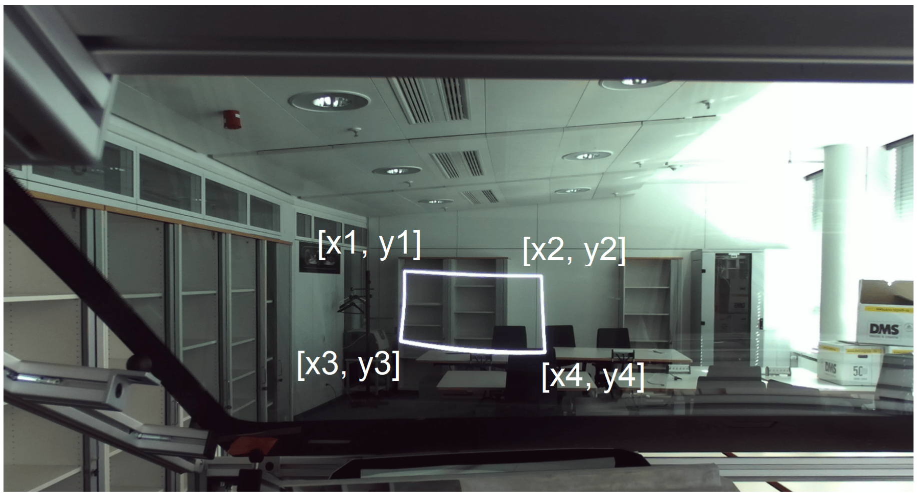
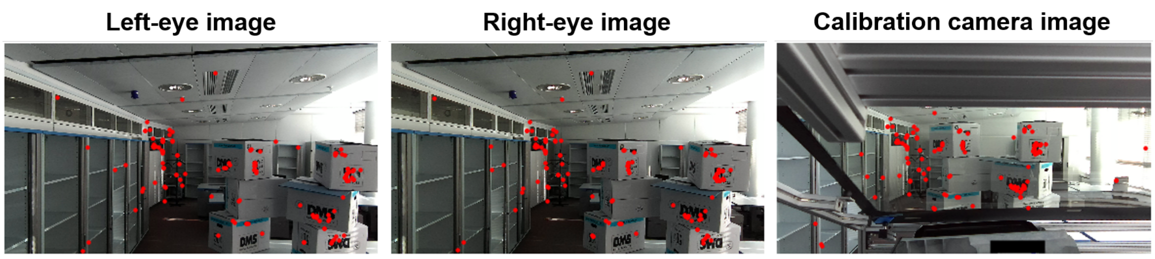


Figure 4.14: Target-free estimation of HUD's focal lengths. The virtual image frame is rendered and captured by the smartphone camera. All the $[x_k, y_k]^T$ ($k \in \{1, 2, 3, 4\}$) represent its corner positions. Note that the expected rectangular virtual image enclosed by the white margins is distorted.

4.5.3.3 Transformation from World to Viewpoint

We determine the transformation $\mathbf{T}_{V_iW}$ from world to viewpoint spaces by exploiting the 2D–2D correspondences among common feature points from the stereo and the smartphone camera images. We take three images of the frontal scene, one with the ZED camera's left eye (denoted with $I_{\text{ZED},L}$), another with its right eye (denoted with $I_{\text{ZED},R}$) and the last one with the smartphone camera (denoted with I_C). Then we use the SIFT algorithms to extract feature points onside the three images, as is introduced in Section 2.7. An example of those feature points is shown in Figure 4.15. We first find matching point-pairs in I_C (point set denoted with p_C) and $I_{\text{ZED},L}$ (point set denoted with p_L). For each feature point $p_{C,j}$, we find the two nearest neighbors by comparing its descriptor with those of all the p_L. Then we perform Lowe's ratio test [51] to ensure the matched pairs are distinctive. That means a point $p_{C,j}$ does not have multiple candidate matching points $p_{L,j}$ whose descriptors are all very similar. After that, we use the same method to find corresponding points (p_R) of the matched points (p_L) in the image $I_{\text{ZED},R}$. Next, we use these extracted correspondences to estimate the fundamental matrices $\mathbf{F}_L$ and $\mathbf{F}_R$ following the theories in Section 2.6:

$$p_C^T \cdot \mathbf{F}_L \cdot p_L = 0 \qquad p_C^T \cdot \mathbf{F}_R \cdot p_R = 0 \tag{4.7}$$



(a) Left-eye image from ZED camera.

(b) Right-eye image from ZED camera.

(c) Image from smartphone camera.

Figure 4.15: Captured images from ZED stereo camera (a–b) and smartphone camera (c) with SIFT-based feature points marked (red circles).

Given that we have pre-calibrated the ZED stereo camera and the smartphone camera, their intrinsic matrices are known, i.e. $\mathbf{K}_L$ for the ZED's left-eye camera, $\mathbf{K}_R$ for the ZED's right-eye camera, and $\mathbf{K}_C$ for the smartphone camera. Therefore, we put them into the calculation of essential matrices:

$$\mathbf{E}_{\text{init},L} = \mathbf{K}_C^T \mathbf{F}_L \mathbf{K}_L \qquad \mathbf{E}_{\text{init},R} = \mathbf{K}_C^T \mathbf{F}_R \mathbf{K}_R \tag{4.8}$$

As is stated in Section 2.6, we can totally derive four candidate transformations $\mathbf{T}_{V_iW}$ for $\pm\mathbf{E}_{\text{init},L}$. Subsequently, we perform the cheirality constraint [49] to find the correct one, with which the number of reconstructed 3D points possessing positive depth values is the largest. Till now, the rotation matrices ($\mathbf{R}_{V_iW}$ and $\mathbf{R}_{V_iW_R}$) and unit translation vectors ($\mathbf{t}_{V_iW}$ and $\mathbf{t}_{V_iW_R}$) are recovered.

However, the scale information of those translation vectors is still missing. To determine the scaling factors, we utilize the pre-calibrated transformation $\mathbf{T}_{WW_R}$ (see also Section 4.1.1) of the stereo camera. We notice the relation that the transformation from the world space W to the viewpoint space V_i should be equivalent to the combination of the following two transformations:

- transformation from the world space W to the right-world space W_R;
- transformation from W_R to the viewpoint space V_i.

This relation is expressed using matrix multiplication:

$$\begin{aligned}\mathbf{T}_{V_iW} &= \begin{bmatrix}\mathbf{R}_{V_iW} & \alpha_1\mathbf{t}_{V_iW}\\ 0 & 1\end{bmatrix}\\ &= \mathbf{T}_{V_iW_R}\mathbf{T}_{WW_R} = \begin{bmatrix}\mathbf{R}_{V_iW_R} & \alpha_2\mathbf{t}_{V_iW_R}\\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{R}_{WW_R} & \mathbf{t}_{WW_R}\\ 0 & 1\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{R}_{V_iW_R}\mathbf{R}_{WW_R} & \mathbf{R}_{V_iW_R}\mathbf{t}_{WW_R}+\alpha_2\mathbf{t}_{V_iW_R}\\ 0 & 1\end{bmatrix},\end{aligned} \tag{4.9}$$

where $\alpha_k > 0$ ($k \in \{1,2\}$) represent the scaling factors to solve. Here, we focus on this equation:

$$\alpha_1\mathbf{t}_{V_iW} = \mathbf{R}_{V_iW_R}\mathbf{t}_{WW_R} + \alpha_2\mathbf{t}_{V_iW_R}. \tag{4.10}$$

We denote $\mathbf{t}_{V_iW} = [t_{1,x}, t_{1,y}, t_{1,z}]^T$, $\mathbf{t}_{V_iW_R} = [t_{2,x}, t_{2,y}, t_{2,z}]^T$ and $\mathbf{R}_{V_iW_R}\mathbf{t}_{WW_R} = [c_1, c_2, c_3]^T$ and the equation becomes:

$$\alpha_1\begin{bmatrix}t_{1,x}\\ t_{1,y}\\ t_{1,z}\end{bmatrix} = \begin{bmatrix}c_1\\ c_2\\ c_3\end{bmatrix} + \alpha_2\begin{bmatrix}t_{2,x}\\ t_{2,y}\\ t_{2,z}\end{bmatrix}. \tag{4.11}$$

By rearranging the variables, we obtain an equation in the form of a linear equation of scaling variables:

$$\begin{bmatrix}t_{1,x} & -t_{2,x} & -c_1\\ t_{1,y} & -t_{2,y} & -c_2\\ t_{1,z} & -t_{2,z} & -c_3\end{bmatrix}\begin{bmatrix}\alpha_1\\ \alpha_2\\ 1\end{bmatrix} = \begin{bmatrix}0\\ 0\\ 0\end{bmatrix}, \tag{4.12}$$

which is rewritten as:

$$\mathbf{A}\cdot\boldsymbol{\alpha} = \mathbf{0}. \tag{4.13}$$

Due to approximation in the calculation and noises in the measurement, it becomes a least-squares problem under the constraints:

- the first two elements of $\boldsymbol{\alpha}$ are greater than zero;
- the last element of $\boldsymbol{\alpha}$ is equal to one.

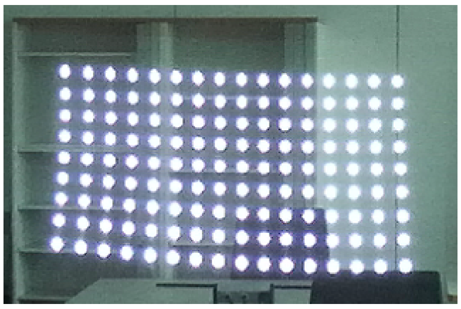
By solving this constrained least-square problem, we recover the scale information of $\mathbf{t}_{V_iW}$ and $\mathbf{t}_{V_iW_R}$. Thus, we have determined the transformation $\mathbf{T}_{V_iW}$ from the world to the viewpoint space.

4.5.3.4 Rotation from Viewpoint to HUD-FOV

The last sub-task is to determine the relative rotation $\mathbf{R}_{H_iV_i}$ from the calibration camera to the HUD-FOV. To this end, we would like to apply the PnP algorithms again. Therefore, we need a group of 2D–3D correspondences from the viewpoint space V_i to the virtual image space I_i. As is mentioned in Section 2.4, we have to find at least three such correspondences. To obtain such correspondences, we first render a 2D virtual point array (Figure 4.16 (a)) whose pixel positions $p_{ij,\mathrm{tr}}$ (or $p_{j,\mathrm{tr}}$ since they can stay the same for each viewpoint) in the virtual image are known, and then extract the centroid of each blob c_{ij} in the captured image at the viewpoint (Figure 4.16 (b)). Next, we construct optical rays $\mathbf{r}_{ij,\mathrm{tr}}$ by back-projecting $c_{ij,\mathrm{tr}}$ into the train viewpoint space $V_{i,\mathrm{tr}}$, as is earlier detailed in Eq. 4.5.

As is mentioned in Section 4.1.3, PnP algorithms should avoid coplanar cases [74], i.e. the 3D control points should not lie on the same plane. Hence on each ray $\mathbf{r}_{ij}$, we select a point with a random depth value in a reasonable range, e.g. from 3 m to 6 m. Indeed, because there is only a relative rotation but no translation between the viewpoint space and HUD-FOV space, the solution of this PnP problem is determined by these optical rays, as is shown in Figure 4.17. Put alternatively, as long as the selected control points stay on the same optical rays, the 2D–3D correspondences lead to the same solution of $\mathbf{R}_{H_iV_i}$. That is why we can randomly assign the depth values under the condition that they are non-coplanar.

With the knowledge of the HUD's intrinsic matrix $\mathbf{K}$, these generated 3D points $P_{ij,\mathrm{tr}}$ and their corresponding 2D points $p_{ij,\mathrm{tr}}$ are then used for the PnP algorithm to calculate the rotation matrix $\mathbf{R}_{H_iV_i}$. At this point, we have recovered the full projection matrix $\mathbf{P}_{I_iW}$ in the pinhole camera model in Eq. (2.5).


(a) Virtual points within the HUD-FOV.

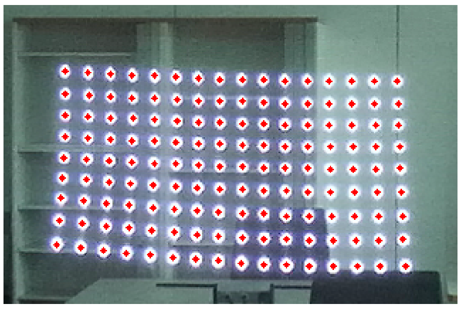
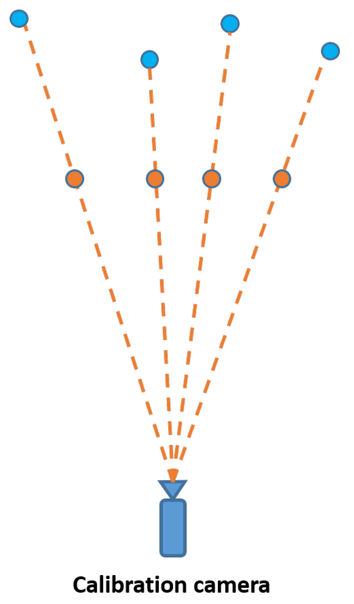
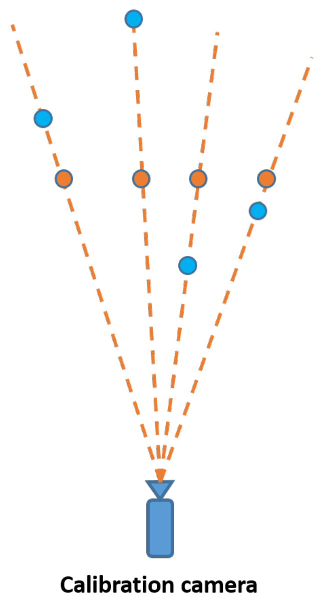

(b) Detected centroids (red dots) of the virtual points.

Figure 4.16: Configured virtual point array $p_{ij,\mathrm{tr}}$ (a) and the recognized centroids $c_{ij,\mathrm{tr}}$ (b). Note that the array is deviated from a rectangular form due to the optical distortion.



(a) A group of 2D–3D correspondences in a viewpoint space $V_{i,\mathrm{tr}}$.



(b) Another group, where 2D points and optical rays remain the same.

Figure 4.17: Configured 2D virtual points and two examples of corresponding assigned 3D control points with random depth values. Orange circles: 2D virtual points $p_{ij,\mathrm{tr}}$; blue circles: 3D control points $P_{ij,\mathrm{tr}}$.

4.5.3.5 Warping Maps & Interpolation

We inherit the tool of warping maps to compensate for the optical distortion. Therefore, the procedure here is identical to that in Section 4.5.2.5. We reproject the above randomly reconstructed control points $P_{ij,\mathrm{tr}}$ back to the virtual image I_i using the solved matrix $\mathbf{P}_{I_iV_i}$ and compare the results $p_{ij,\mathrm{va}}$ with the input array $p_{ij,\mathrm{tr}}$. The differences, i.e. $p_{ij,\mathrm{tr}} - p_{ij,\mathrm{va}}$, are regarded as bias vectors, from which we reconstruct warping maps for both Δu– and Δv–components. To deal with uncalibrated parameters at non-participating viewpoints, we interpolate both the linear matrices ($\mathbf{T}_{V_iW}$ and $\mathbf{R}_{H_iV_i}$), and the warping maps acquired from the train viewpoints bilinearly. Thus, the perspective projection and distortion model at any viewpoint inside the eye box are restored.

4.5.3.6 Summary

Figure 4.18 summarizes our target-free AR-HUD calibration scheme. Instead of using any pre-designed calibration target, we directly extract adequate feature points in the frontal scene and apply the epipolar constraint to recover the relative poses between the 3D sensor and the calibration camera. The estimations of the HUD projector's intrinsic parameters and the relative rotation from viewpoint space to the HUD-FOV are also accomplished without any target. The initial calibration phase including 9 train viewpoints takes less than 5 min, including the initialization of our devices. The feature points can be out of the narrow HUD-FOV - they are only required to lie in the joint FOV of our 3D sensor and calibration camera. Hence, they are readily sampled by our cameras and home-made software.

Significantly, our target-free approach is promising for customers to calibrate AR-HUDs themselves because the efforts in preparing targets are circumvented. When corresponding on-vehicle connections and interfaces are developed, the drivers can park their cars in front of a scene with enough textures and stable illumination (at least within a few minutes), such as a building or a gate, and then start the calibration on their smartphones at the accustomed eye positions. Thus, a visit to the factory or workshop becomes unnecessary, saving relevant costs and human labor.

We also notice that the frontal scene's quality may affect the recovered transformation part $\mathbf{T}_{V_iW}$. Compared to the target-based rivals, the current implementation may be more sensitive to noise in feature extraction, unfavorable illumination or other adverse conditions, which are the anticipated challenges in various scenarios.

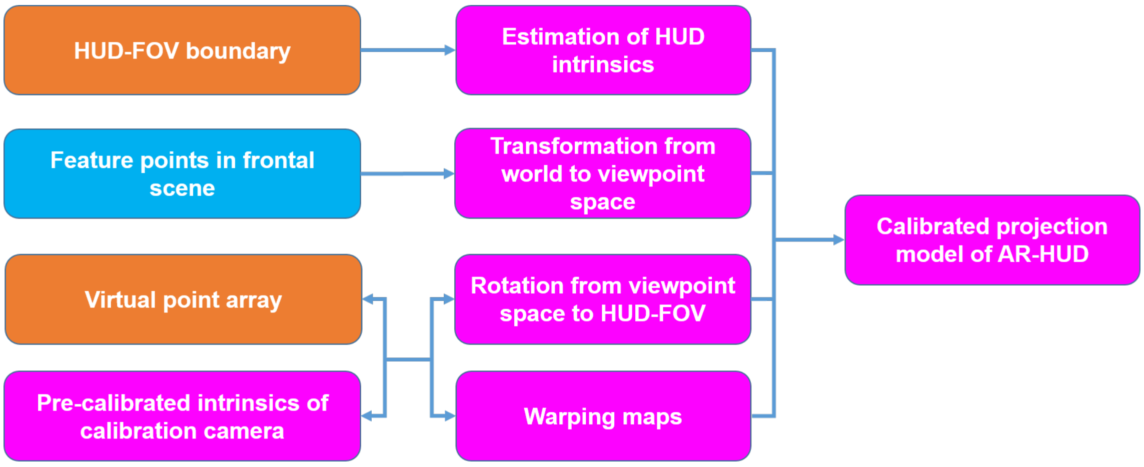

HUD-FOV boundary
Estimation of HUD intrinsics
Feature points in frontal scene
Transformation from world to viewpoint space
Calibrated projection model of AR-HUD
Virtual point array
Rotation from viewpoint space to HUD-FOV
Pre-calibrated intrinsics of calibration camera
Warping maps

Figure 4.18: Target-free calibration pipeline. Different colors label different data types: blue for 3D positions, orange for 2D positions and magenta for acquired data.

5. Results

This chapter encompasses all our simulated and experimental results, particularly those acquired data under different calibration concepts. Before showing the results in detail, we explain our criteria on relevant evaluation and statistics in Section 5.1. Next, Section 5.2 shows the simulations under various assumptions, through which we can build up a deeper understanding of the possible influencing factors in the AR-HUD calibration. Section 5.3 provides readers the pre-calibration results of all the connected sensors, including different smartphone cameras and ZED stereo cameras. In Section 5.4 and 5.5, we sequentially present the obtained calibration results in the manual and automatic implementations, corresponding to the experimental schemes introduced in Section 4.4 and 4.5.

5.1 Criteria on Projection Results

We have to unify the quantitative criteria to demonstrate the reprojection or projection errors in the validation and evaluation phases. We calculate them as the root-mean-square error (RMSE). For example, when we have N 2D–3D correspondences and M viewpoints, in the validation phase, we have the reprojection error in pixel as:

$$\mathrm{RMSE}_{\mathrm{va,px}} = \frac{\sum_{i=1}^{M} \sqrt{\frac{\sum_{j=1}^{N}\left[(u_{ij,\mathrm{va}}-u_{ij,\mathrm{tr}})^2+(v_{ij,\mathrm{va}}-v_{ij,\mathrm{tr}})^2\right]}{N}}}{M}, \tag{5.1}$$

where $p_{ij,\mathrm{va}} = [u_{ij,\mathrm{va}}, v_{ij,\mathrm{va}}]^T$ and $p_{ij,\mathrm{tr}} = [u_{ij,\mathrm{tr}}, v_{ij,\mathrm{tr}}]^T$ are reprojected and ground truth virtual points, respecitvely. Since we have estimated our HUD's focal lengths, i.e. f_u and f_v, then we convey this RMSE in pixel to that in physical length at a distance d_{stat} from the eye box:

$$\mathrm{RMSE}_{\mathrm{va}} = \frac{\mathrm{RMSE}_{\mathrm{va,px}}}{\bar{f}_{u,v}} \cdot d_{\mathrm{stat}}, \tag{5.2}$$

where $\bar{f}_{u,v} = (f_u + f_v)/2$. In the evaluation phase with test sets, we calculate a likewise the projection error between the projected ($\hat{p}_{ij,\mathrm{te}}$) and ground truth ($p_{ij,\mathrm{te}}$) virtual points.

The above criteria are built upon pixel positions in the virtual image. Similarly, if we use captured photos from the calibration camera for the quantitative evaluation, then we have:

$$\mathrm{RMSE}_{\mathrm{te,px}} = \frac{\sum_{i=1}^{M} \sqrt{\frac{\sum_{j=1}^{N}\left[(\hat{x}_{ij,\mathrm{te}}-x_{ij,\mathrm{te}})^2+(\hat{y}_{ij,\mathrm{te}}-y_{ij,\mathrm{te}})^2\right]}{N}}}{M} \tag{5.3}$$

and

$$\mathrm{RMSE}_{\mathrm{te}} = \frac{\mathrm{RMSE}_{\mathrm{te,px}}}{\bar{f}_{x,y}} \cdot d_{\mathrm{stat}}, \tag{5.4}$$

where $\hat{c}_{ij,\text{te}} = [\hat{x}_{ij,\text{te}}, \hat{y}_{ij,\text{te}}]^T$ represent the projected virtual points' centroids in the photos, and $c_{ij,\text{te}} = [x_{ij,\text{te}}, y_{ij,\text{te}}]^T$ are the captured features to be augmented. The term $\bar{f}_{x,y}$ is the averaged focal lengths in pixel of the calibration camera with $\bar{f}_{x,y} = (f_x + f_y)/2$.

The standard deviation (STD) is calculated for distribution of all the individual (re)projection error at the distance d_stat. An individual error is expressed like:

$$e_{ij,\text{va}} = \frac{\sqrt{\left[(u_{ij,\text{va}} - u_{ij,\text{tr}})^2 + (v_{ij,\text{va}} - v_{ij,\text{tr}})^2\right]}}{\bar{f}_{u,v}} \cdot d_\text{stat} \tag{5.5}$$

w.r.t. the virtual image for the validation phase or

$$e_{ij,\text{te}} = \frac{\sqrt{\left[(\hat{x}_{ij,\text{te}} - x_{ij,\text{te}})^2 + (\hat{y}_{ij,\text{te}} - y_{ij,\text{te}})^2\right]}}{\bar{f}_{x,y}} \cdot d_\text{stat} \tag{5.6}$$

w.r.t. the photos from the calibration camera for the evaluation phase.

For a clear comparison, we always take $d_\text{stat} = 7.5\,\text{m}$, the same as in state of the art [10] that provided quantitative evaluation.

5.2 Simulation Results

In the simulation, we build up a virtual HUD system and calibration environment. The main tasks are to investigate our physical model's validity and determine the influence of possible error sources, including the 3D sensing and 2D detection errors, as well as optical distortion. We also consider the robustness of calibration under different restrictions, such as the number of 2D–3D correspondences. Although our simulation differs from the practical experiment in several aspects, we can still draw some useful conclusions as guidance for our implementations. Nevertheless, the simulation results can also be valuable for reference purposes, e.g. for a robust design of automotive AR-HUD systems.

All the simulations are carried out with our home-made *Python* (Release 3.7.0) code. We separate various kinds of error sources and add them individually in programming to reveal every single sort's influence. In most of these sub-tasks, we set the number of iteration as 1000 to acquire an average effect in case of possible outliers. Note that the "mean" errors in our simulations means simply the RMSE averaged over the number of iteration.

5.2.1 Error-free AR-HUD System

An error-free AR-HUD is an ideal system that escapes any systematic or random error. That means we ignore any 3D or 2D detection error, as well as any optical distortion. Although this is not a realistic case, it is a primary step to validate our calibration concept based on the pinhole camera model in Section 2.4. We can examine the recovery of relevant matrices and parameters using precise 2D–3D correspondences and PnP algorithms.

If it works, then the linear part, i.e. the obtained perspective projection matrix $\mathbf{P}_{I_iW}$ in Eq. (2.5)–(2.9), should be able to approximate the imaging behavior in AR-HUDs. Furthermore, for the following cases with error sources, we can still execute the simulations similarly.

Since this virtual AR-HUD is an ideal model, we simulate the calibration at 25 viewpoints (9 as train and 16 as test set), which are chosen the same as shown in the eye box in Figure 3.2 (a). We first deal with the calibration and validation at the train viewpoints. The pre-defined intrinsic matrix $\mathbf{K}$ remains the same throughout the simulation. For a train viewpoint $V_{i,\mathrm{tr}}$, we pre-define a transformation from the world W to it, then based on the eye box's geometry, we know the corresponding ground truth extrinsic matrices $\mathbf{T}_{H_iW}$ for all the viewpoints. When we start any iteration of the simulation, for each viewpoint, we randomly set 50 non-coplanar 3D control points P_{ij} in the world space W, as is shown in Figure 5.1. Then at each viewpoint, we use the intrinsic matrix $\mathbf{K}$ and its extrinsic matrix $\mathbf{T}_{H_iW}$ to project P_{ij} onto the 2D virtual points $p_{ij,\mathrm{tr}}$ for the train set or $p_{ij,\mathrm{te}}$ for the test set. Thus, we have 50 2D–3D correspondences. Above is a forward step to construct the ground truth perspective projections, which are error-free.

Next, we do the backward step using the known $\mathbf{K}$, and 2D–3D correspondences between $p_{ij,\mathrm{tr}}$ and P_{ij}. This is, actually, the simulated extrinsic calibrations at those $V_{i,\mathrm{tr}}$, followed by a qualitative validation. We solve the extrinsic matrices $\hat{\mathbf{T}}_{H_iW}$ by employing the PnP algorithms. Then we compare them with the ground truth $\mathbf{T}_{H_iW}$ elementwise. Meanwhile, we also compare these two groups by doing reprojections, which means we compute the validation set of virtual points $p_{ij,\mathrm{va}}$ using $\mathbf{K}$, $\hat{\mathbf{T}}_{H_iW}$ and P_{ij}. We further compare the biases between the ground truth p_{ij} and reprojected $p_{ij,\mathrm{va}}$.

After the validation, we will do the evaluation at test viewpoints $V_{i,\mathrm{te}}$ to examine the interpolation concept described in Section 4.1.4 and 4.5.1.5. As is stated, we convert each calculated extrinsic matrix $\hat{\mathbf{T}}_{H_iW}$ into a Rodrigues rotation vector $\hat{\mathbf{r}}_{H_iW}$ and a translation vector $\hat{\mathbf{t}}_{H_iW}$. We interpolate these vectors for all the test viewpoints bilinearly, and convert the interpolated results back to transformation matrices. Using them, we reproject the 3D control points P_{ij} to the corresponding virtual image space I_i, and compare the acquired virtual points $\hat{p}_{ij,\mathrm{te}}$ with the ground truth $p_{ij,\mathrm{te}}$.

Table 5.1 shows the ground truth and recovered extrinsic parameters at an example train viewpoint and then a test viewpoint. Without any error source, we can see that the calibrated ones are identical to the ground truth. Figure 5.2 shows the simulated validation result at a train viewpoint and the evaluation result at a test one, which demonstrates accurate overlapping between ground truth and (re)projected virtual points. Therefore, it is validated that our calibration pipeline works well in estimating the linear perspective projection part if the sensors' detection accuracy is guaranteed and the optical distortion is negligible.

5.2.2 AR-HUD System with 3D Detection Error

There are mainly three different sorts of possible detection errors in our stereo camera, even after pre-calibration. One is the systematical error that turns out absolute biased

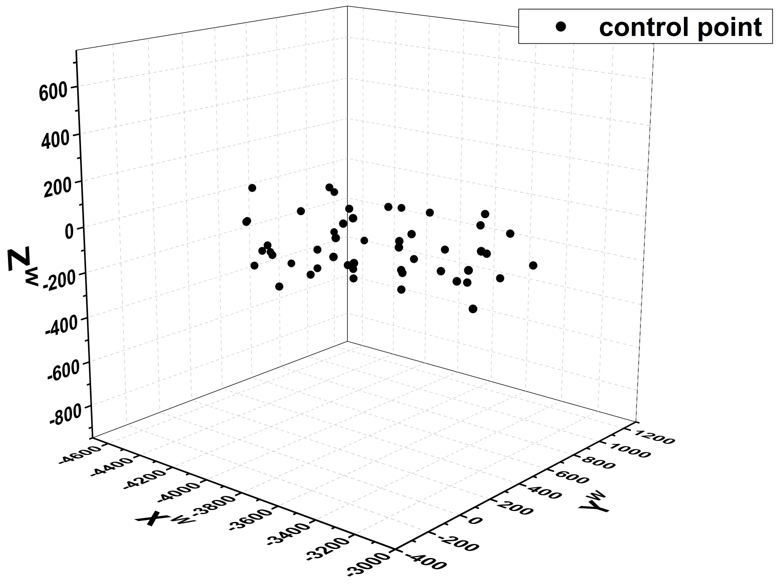
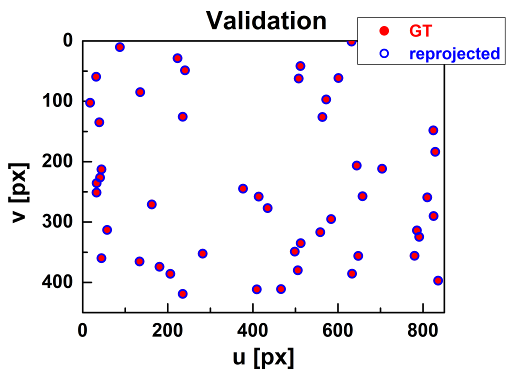
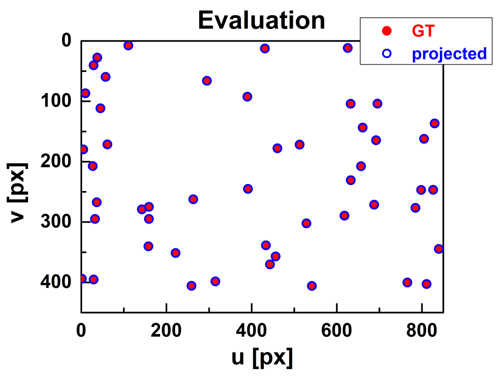


Figure 5.1: 50 simulated 3D control points P_{ij} in the world space W for a viewpoint $V_{i,\mathrm{tr}}$ in an iteration. They are randomly chosen with a depth range of 3.5 m to 5.0 m from the eye box.



(a) Validation using 50 random control points at a train viewpoint $V_{i,\mathrm{tr}}$.



(b) Evaluation using another 50 random control points at a test viewpoint $V_{i,\mathrm{te}}$.

Figure 5.2: Simulated (re)projection results after the calibration of an ideal HUD. GT: ground truth; px: pixel.

Table 5.1: Calibrated prameters (Rodrigues rotation vector $\mathbf{r} = [r_1, r_2, r_3]^T$ and translation vector $\mathbf{t} = [t_1, t_2, t_3]^T$) of an ideal AR-HUD system at a train viewpoint and evaluation at a test viewpoint, regardless of any error source. GT: ground truth.

Element	Train viewpoint (GT)	Train viewpoint (PnP)	Relative error (%)
r_1	1.24546	1.24546	0
r_2	1.22819	1.22819	0
r_3	-1.17985	-1.17985	0
t_1	474.02	474.02	0
t_2	-154.291	-154.291	0
t_3	425.771	425.771	0

Element	Test viewpoint (GT)	Test viewpoint (PnP)	Relative error (%)
r_1	1.24546	1.24546	0
r_2	1.22819	1.22819	0
r_3	-1.17985	-1.17985	0
t_1	494.018	494.018	0
t_2	-134.319	-134.319	0
t_3	426.853	426.853	0

detected 3D positions; the other two, i.e. the disparity error and the pattern recognition error, are random errors in real-time detection. Though all these errors may appear in the 3D sensing, they have different influences on the calibration results. Errors from other 3D sensors, such as the RGB-D camera or the LiDAR, can be modeled likewise, as long as their respective characters are taken into account.

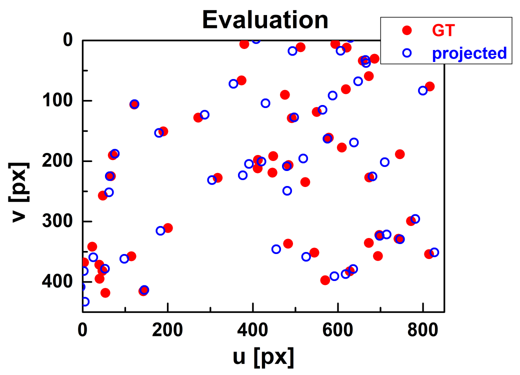
5.2.2.1 Systematical Error

The systematical error means that there always exists a constant bias $\left[\Delta X^W, \Delta Y^W, \Delta Z^W\right]$ in the 3D detection. Alternatively speaking, the 3D sensor is not precisely aligned with the defined world or vehicle space W but stays stable. This bias will lead to biased extrinsic matrices $\mathbf{T}_{H_iW}$ for each viewpoint. However, theoretically, the bias for each viewpoint should remain the same if the sensor remains fixed during the implementation.

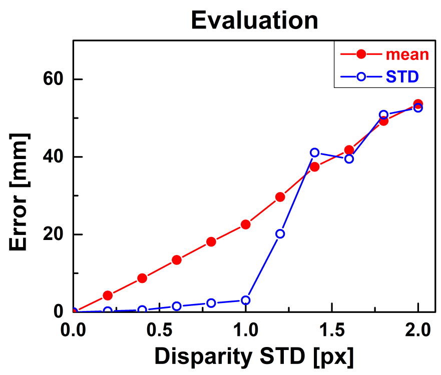
Thus, if the 3D sensor remains unchanged as in the AR-HUD calibration, the presented virtual images should not be affected when presenting any AR content. This is because the biased extrinsic calibration and the biased projection using the solved $\mathbf{T}_{H_iW}$ form a close loop, or rather, the biases from these two procedures cancel each other. Nevertheless, the constant bias may lead to issues when the extrinsic parameters are called in additional routines that are indirectly related to AR-HUD calibration.

5.2.2.2 Disparity Error

The disparity or depth error refers to the deviation in the 3D sensing to acquire the depth value (Z^W) of a detected real-world position, e.g. that in pixel triangulation. Here we add on some noise in the simulated 3D sensing. As is expressed in Eq. 2.14, since the detected depths are derived from disparity values, we assume that there are Gaussian random errors in them. We tune the standard deviation of this Gaussian distribution, calibrate the virtual AR-HUD, and finally investigate the changing of projection error in the evaluation phase. The results are plotted in Figure 5.3. As we notice in Figure 5.3 (a), under the influence of fluctuation in the disparity, the pixel positions of those reprojected virtual points deviate from the ground truth. We also plot the average reprojection errors and their corresponding standard deviations as a function of the disparity errors' variance in Figure 5.3 (b). The average reprojection error increases almost linearly. However, its standard deviation shows a jumping behavior when the disparity error fluctuates to a certain level.



(a) Projection result when $\sigma_d = 2.0\,\text{px}$.



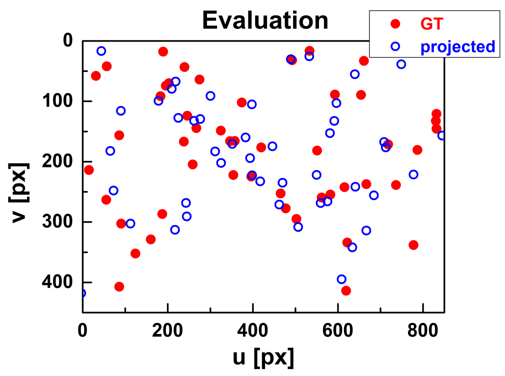
(b) Statistics for a series of σ_d.

Figure 5.3: Simulation results with disparity errors in the 3D sensing. Here we have a total of 50 random 2D–3D correspondences inside the HUD-FOV. GT: ground truth; STD: standard deviation; px: pixel; σ_d: standard deviation of Gaussian errors added on disparity.

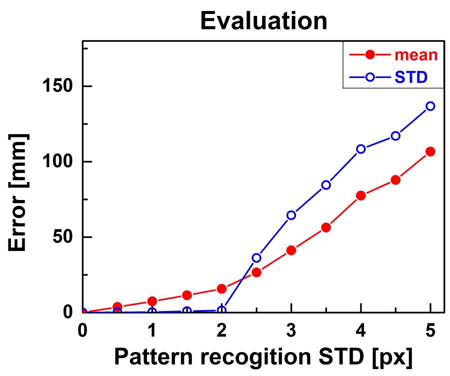
5.2.2.3 Pattern Recognition Error

The pattern recognition error refers to the biased 3D detection of pre-selected control points, mainly when we apply a designed calibration target. For example, in our case, even though there is no disparity error, there exists the chance that the stereo camera cannot locate the features on the target correctly. This means there might be some biases between the ground truth pixel position $[x_i, y_i]^T$ and the recognized pixel position $[x'_i, y'_i]^T$ in the captured image pairs of the stereo camera. Here we add a series of normally distributed biases $[\Delta x_i, \Delta y_i]^T$ whose mean values are set as zero, but standard deviations vary. This is not a systematic error within the stereo camera because this error changes at different detectable control points.

A corresponding evaluation example is shown in Figure 5.4 (a), where the configured standard deviation is 5.0 pixel, while Figure 5.4 (b) plots the statistics with increasing fluctuation in pattern recognition. In our simulation environment, it is best to suppress the standard deviation of pattern recognition error down to 2 pixels under the assumption that the stereo camera has a Full HD resolution. On the curve (the blue one) of standard deviations in the reprojection errors, we notice again a jumping of the slope, which is similar to Figure 5.3 (b).



(a) Projection result when $\sigma_{xy} = 5.0\,\text{px}$.



(b) Statistics for a series of σ_{xy}.

Figure 5.4: Simulation results with pattern recognition errors in the 3D sensing. Here we have a total of 50 random 2D–3D correspondences inside the HUD-FOV. GT: ground truth; STD: standard deviation; px: pixel; σ_{xy}: standard deviation of a single set of pattern recognition errors.

5.2.2.4 Summary

In general, the absolute accuracy and stability of the 3D sensing matter much in our calibration precision. Since the calibration of each vehicle's AR-HUD is a one-time effort in the workshop, the fluctuation in the 3D sensing should be restricted within a satisfying interval. However, this requirement depends not only on the sensors' quality: we have to guarantee that the environment should be experimentally favorable for robust detection. For example, the illumination should be appropriately controlled, or the features and control points should be enough and properly distributed to suppress potential disparity or pattern recognition error.

5.2.3 AR-HUD System with 2D Detection Error

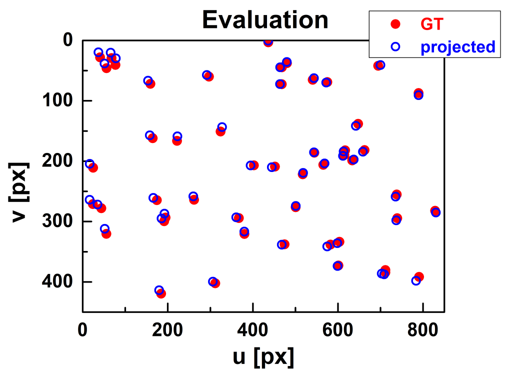
The 2D detection error mainly refers to the deviated detection of virtual points in the automatic calibration schemes. Sometimes, it also happens in the manual implementation, e.g. considering the operator's visual fatigue. When the pre-calibration of the calibration camera is finished (see Section 4.1.1), we can assume no systematical error in the 2D detection but only random errors. Moreover, we have to concern the so-called "pixel quantization" effect because in both the virtual images and captured images from the smartphone, the acquired pixel values can only be rounded to integers, though 2D–3D matchings (see Section 4.4.1 and 4.5.1) or 2D virtual point localization (see 4.5.2 and 4.5.3) may appear at non-integer pixel positions.

5.2.3.1 Random Error

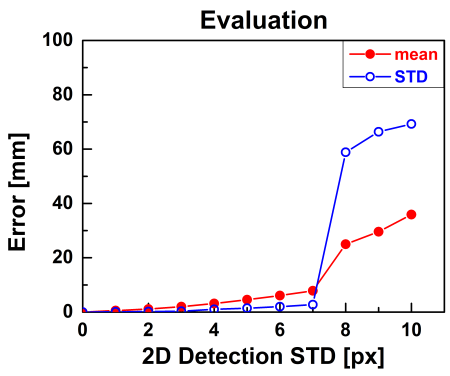
The random 2D detection errors lead to the incorrect overlapping between the virtual point array and the 3D control points, or inaccurate localization of virtual points in the calibration camera's view. In our implementations, it occurs due to careless manual shifting or imperfect contour extraction algorithms. Here we add a series of Gaussian noise $[\Delta u_{ij}, \Delta v_{ij}]^T$ in this process, whose mean value is zeros but standard deviation increases.

After simulated calibration with random 2D detection error, we demonstrate an evaluation example in Figure 5.5 (a), where the pre-set standard deviation is 10.0 pixel. The projected virtual points are generally located close to the ground truth, but apparently, there exist differences between their pixel positions in the virtual image I_i. Figure 5.5 (b) shows the statistics with increasing standard deviation in the Gaussian noise. A jumping behavior happens on both the curves of mean projection errors and their standard deviations in the interval from 7.0 pixels to 8.0 pixels.

As a matter of fact, this error's influence is related to the virtual image's resolution. For example, a virtual image with a 640×480 pixel resolution should be more vulnerable to a certain amount of 2D detection biases than a Full HD one. Nevertheless, this error should be suppressed at least within the limit over which the jumping of projection errors occurs; otherwise, a one-time calibration with a large enough fluctuation in the 2D detection performance may lead to outlier parameters.



(a) Projection result when $\sigma_{uv} = 10.0$ px.



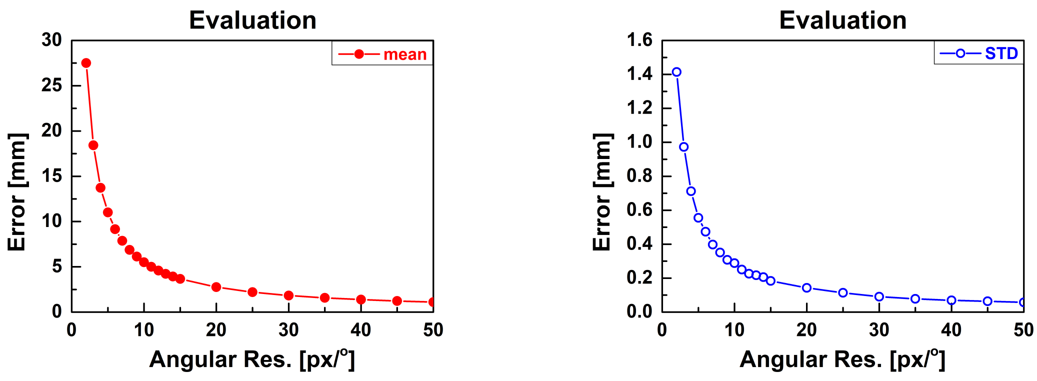
(b) Statistics for a series of σ_{uv}.

Figure 5.5: Simulation results with 2D detection errors. Here we have a total of 50 random 2D–3D correspondences inside the HUD-FOV. GT: ground truth; STD: standard deviation; px: pixel; σ_{uv}: standard deviation of a single set of 2D detection errors.

5.2.3.2 Pixel Quantization Error

Another 2D detection error that we have to consider is the pixel quantization error. This error comes from the mismatching between the resolutions of the calibration camera and the virtual image. For example, when we use a calibration camera with 1920×1080 pixel resolution, whereas the virtual image has a size of 800×600, then a pixel in the virtual image can cover different neighboring pixels in the captured image or vice versa. Even if they have the same resolution, the relative pose between their image planes can still lead to this error. However, the pixel values we acquired are always rounded to integers. Besides, when any 3D real-world position P_j is projected to the virtual image I_i, the raw projection $p_{ij} = [u_{ij}, v_{ij}]^T$ is also rounded to an integer pixel location $\hat{p}_{ij} = [\hat{u_{ij}}, \hat{v_{ij}}]^T$. A single rounding can lead to maximum matching error of $0.5 \times \sqrt{2} \approx 0.7$ pixel.

As is mentioned in Section 5.1, the absolute projection error in length at a certain distance depends on two factors, i.e. the HUD-FOV (opening angles) and the virtual image resolution. Here we fix the HUD-FOV to 40° × 20° and check the influence of the angular virtual image resolution, i.e. pixel per degree. Other influences of opening angles are detailed later in Section 5.2.7. In the current simulation, other error sources are removed except for the pixel quantization. The statistics of simulated average projection errors and their standard deviations are plotted in Figure 5.6. As we notice, both the errors and their spreading fall monotonous down to a convergent level close to zero. This means for HUDs with a fixed FOV, a larger angular resolution helps accomplish a more precise calibration. However, this number is highly dependent on the manufacturing of the HUD projector and potential costs.



(a) Average projection error against angular resolution.

(b) Standard deviation of projection errors against the angular resolution.

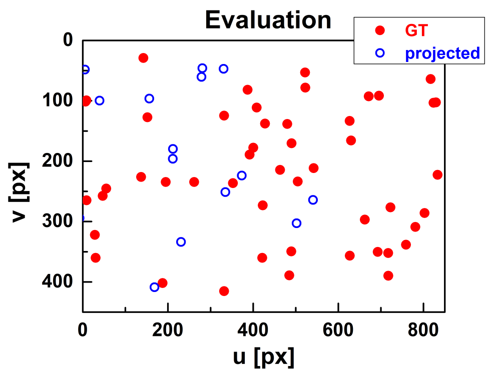
Figure 5.6: Simulation results with pixel quantization errors. GT: ground truth; STD: standard deviation; Res.: resolution; px/°: pixel per degree.

5.2.4 Number of 2D–3D Point Correspondences

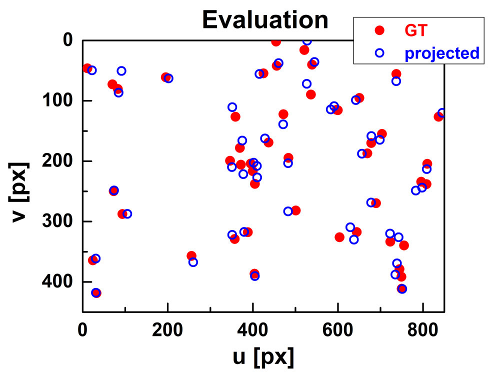
The number of available 2D–3D correspondences between the virtual and control points can affect our AR-HUD calibration accuracy. In Section 2.4.4, we have mentioned that if we solve the extrinsic matrix $\mathbf{T}_{H_iW}$ using PnP algorithms, we need at least three 2D–3D correspondences, i.e. in a P3P form. Other algorithms also have such a requirement. For example, the DLT algorithm [37] requires at least six 2D–3D corresponding pairs because the projection matrix $\mathbf{P}_{I_iW}$ is a 3×4 matrix with 11 DoF. Nevertheless, after all, these are the lower bounds of the required numbers, i.e. the theoretical minimum requirement. We can imagine that too few 2D–3D correspondences may lead to apparent biased calibration results due to various error sources.

Here we investigate this issue using a simple case, where we only add a constant pattern recognition error ($\sigma_{xy} = 3.0$ pixels) in the 3D detection. Note that we have to include an error source; otherwise, the projection errors in the evaluation phase keep zero with an error-free AR-HUD system. We set different numbers of control points and observe the performance. The results are shown in Figure 5.7 and 5.8. If we compare the case with only 6 control points (Figure 5.7 (a)) with the case with 50 ones (Figure 5.7 (b)), we see that the alignment between the projected virtual points and the ground truth is far more precise in the latter than in the former. This is in line with what we have expected.

From the statistical results in Figure 5.8 (a), it is noticed that when the control points reach a threshold number (e.g. over 60), the average projection errors in the evaluation converge to a specific level. On the plotting of the standard deviation in Figure 5.8 (b), a similar convergence appears. The simulated phenomenon denotes that we have to ensure a large enough group of 3D control points so that the influence of errors can be suppressed below a certain level. The main reason is that in the workshop or factory, we cannot sacrifice the costs and efficiency to iterate the calibration routines many times. Especially, this number can be much larger than the theoretical minimum requirements for those calibration algorithms.

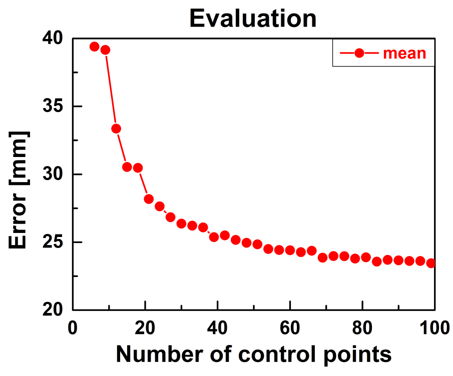


(a) Projection result when 6 control points are used for calibration.

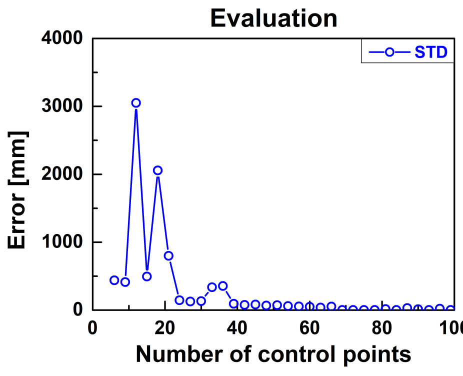


(b) Projection result when 50 control points are used for calibration.

Figure 5.7: Example of simulated evaluation results with various numbers of 2D–3D correspondences inside the HUD-FOV for the calibration. GT: ground truth; px: pixel.



(a) Average projection errors for a series of numbers of control points.



(b) STD of projection errors for a series of numbers of control points.

Figure 5.8: Statistics of simulated evaluation results with various numbers of 2D–3D correspondences inside the HUD-FOV. STD: standard deviation; px: pixel.

5.2.5 Tolerance of Viewpoint Positions

As is introduced in Section 3.6, because the alignment between the 2D virtual image and the 3D real world is highly dependent on the viewpoint, we have to include a driver camera in the vehicle to track its real-time position. This tracker is also deployed in our calibration routine since we calibrate the AR-HUD at multiple viewpoints inside the defined eye box. However, we have to be careful that such a measurement is not error-free. Therefore, we would like to take certain tolerances into account. This consideration mainly comes from two aspects. Firstly, the driver camera may also contain detection errors. In practice, it is not guaranteed that it provides perfect sensing results, particularly concerning the environmental factors, e.g. illumination. Secondly, in the manual implementation using human eyes, since the operator's head cannot be fixed strictly, we have to set an interval where the operator's eyes are "relatively free" to move. However, this interval may also sacrifice the final calibration results. Figure 5.9 shows a 3D plot of 100 various viewpoint positions in the world space W. These viewpoints are centered around a mean position $\left[\bar{X}^W, \bar{Y}^W, \bar{Z}^W\right]^T$, obeying a 3D Gaussian distribution confined by $[\sigma_{X^W}, \sigma_{Y^W}, \sigma_{Z^W}]^T$.

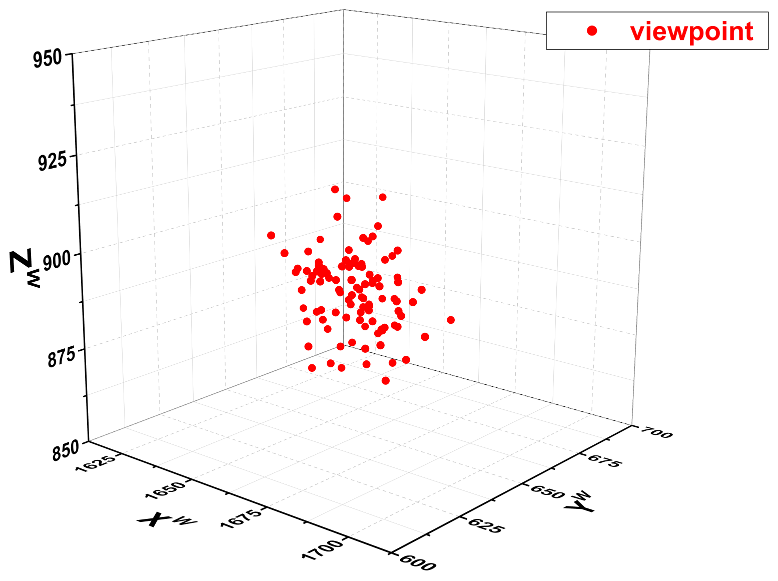


Figure 5.9: 100 simulated various viewpoint positions in the world space W. Here we set $\sigma_{X^W} = 10\,\text{mm}$, $\sigma_{Y^W} = 10\,\text{mm}$ and $\sigma_{Z^W} = 10\,\text{mm}$.

In the simulation, we set σ_{X^W}, σ_{Y^W} and σ_{Z^W} independently as a linear space varying from 0 mm to 100 mm. The simulation results are plotted in Figure 5.10. The variance in each of X^W–, Y^W– and Z^W–axes leads to biases in the evaluation. Nevertheless, it is noticed that the viewpoint displacement in the X^W–direction demonstrates an order smaller contribution to the projection errors than in the other two. The reason is that the opening angles of our AR-HUD are relatively narrow. Therefore, in practice, we only need to define a 2D eye box in the $Y^W Z^W$–plane, as is stated in Section 3.2. However, when the opening angles become larger, the viewpoint displacement in the X^W–axis may also influence the projection results, as is later demonstrated in Section 5.2.7.

We also notice that in Figure 5.10 (c) and (e), the viewpoint's displacement on the Y^W–axis only results in a shift of the u–component of the virtual image, while the that on the Z^W–axis only results in a shift of the v–component. This is intuitively true but may be invalid for a real AR-HUD because the virtual image might have a non-zero roll rotation angle around the X^W–axis. Furthermore, the tolerance of viewpoints on an individual axis affects the final projection errors almost linearly, as is shown in Figure 5.10 (b), (d) and (f).

5.2.6 Optical Distortion

In this simulation, we examine our distortion correction based on warping maps. Without generality, we add random warping bias vectors $\mathbf{v}_{\text{bias}}$ on the ground truth virtual points from an error-free AR-HUD, which corresponds to the train viewpoints in the eye box. Then for the test viewpoints, we can calculate their ground truth bias vectors using bilinear interpolation among train viewpoints.

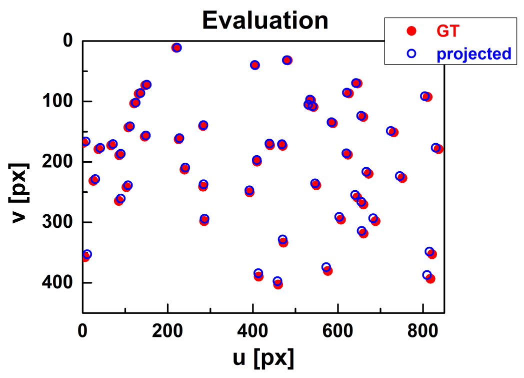
Next, we calibrate the extrinsic matrices $\mathbf{T}_{H_iW}$ for the train viewpoints and then acquire the interpolated ones at the test viewpoints. Meanwhile, in the validation phase, we reconstruct the warping maps for the train viewpoints and interpolate them for those test viewpoints. Finally, we project control points to the virtual image in the test views and rectify them with the interpolated warping maps. Then we can observe the evaluation results qualitatively, as is shown in Figure 5.11. In this example, we set the amplitude of added bias vectors $\|\mathbf{v}_{\text{bias}}\|$ evenly distributed from 0 to 10 pixel and their orientations are evenly distributed in 360°. We observe that the dewarped virtual points are located closer to the ground truth than the raw projected ones.

We change the maximal amplitudes of bias vectors and repeat the simulation. Figure 5.12 shows the statistics. Note that both the projection error and standard deviation fall downwards compared to those in the raw evaluations.

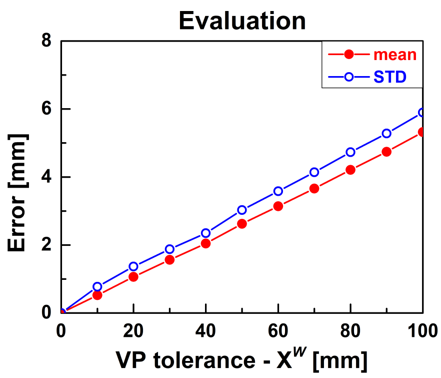
5.2.7 Opening Angles

As is mentioned in Section 3.2, the opening angles of currently available AR-HUDs on the market are still narrower than desired. However, we cannot exclude new AR-HUDs with larger FOV in the future. If the laser-based or waveguide-based AR-HUDs come into mass production and intensive use, we can expect a much more expanded frustum, which is expected to be greater than 40°. In such a case, more elements can be added to the broader virtual image, and the calibration may assert higher requirements. Therefore, it makes sense if we investigate the influence of different opening angles on the final results.

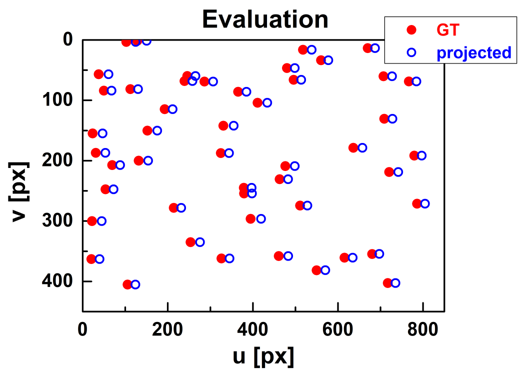
Especially, we have an interest in seeing their relations to the projection errors with a fixed tolerance of the viewpoint's displacement in the X^W–direction. We have assumed this tolerance has little impact on the calibration when the HUD-FOV is narrow, but we were unsure what happens when it became larger. Here we set $\sigma_{X^W} = 100\,\text{mm}$ and the angular resolution (pixels per degree) constant. We use 100 control points and enumerate the horizontal opening angle from 5° to 60°. To keep the aspect ratio, we always set the vertical opening angle equal to half of the horizontal one. The results are shown in Figure 5.13. Since we have expanded the opening angle to 60° while keeping the angular resolution, the virtual image's absolute resolution is also enlarged, as is plotted in Figure 5.13 (a). If we zoom in on the figure, we observe that the biases are smaller in the central region and larger in the periphery. This coincides with the assumption that when the HUD-FOV



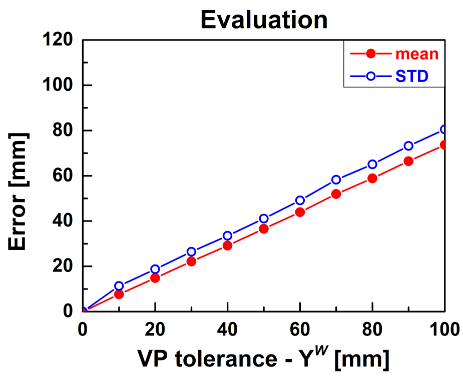
(a) An evaluation example with a tolerance of viewpoint displacement on the X^W–axis with $\sigma_{X^W} = 100\,\mathrm{mm}$.



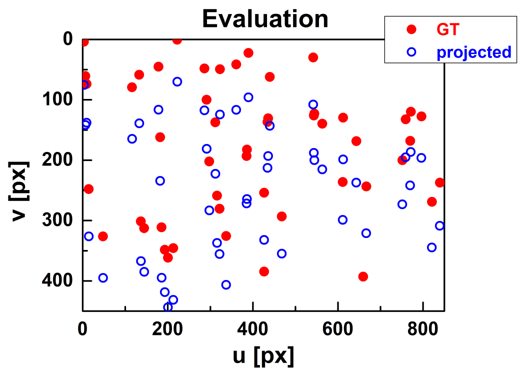
(b) Statistics of simulation results with a series of tolerances of viewpoint displacement on the X^W axis.



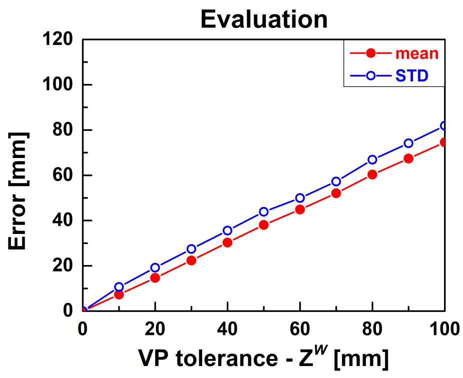
(c) An evaluation example with a tolerance of viewpoint displacement on the Y^W–axis with $\sigma_{Y^W} = 100\,\mathrm{mm}$.



(d) Statistics of simulation results with a series of tolerances of viewpoint displacement on the Y^W axis.

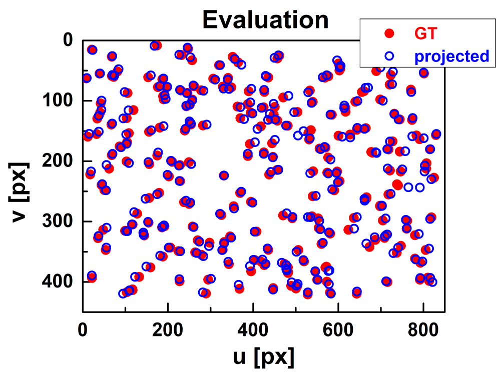


(e) An evaluation example with a tolerance of viewpoint displacement on the Z^W–axis with $\sigma_{Z^W} = 100\,\mathrm{mm}$.

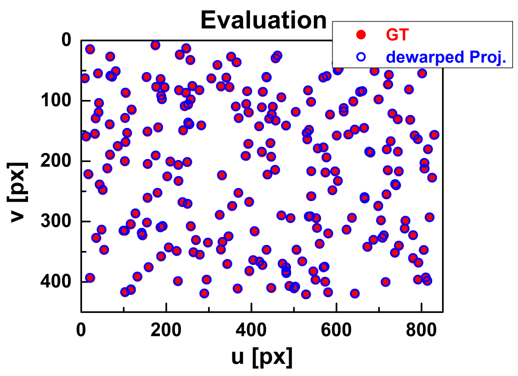


(f) Statistics of simulation results with a series of tolerances of viewpoint displacement on the Z^W axis.

Figure 5.10: Separate simulation results on tolerances of viewpoint positions along the X^W–, Y^W– and Z^W–axis. VP: viewpoint; GT: ground truth; px: pixel; STD: standard deviation.

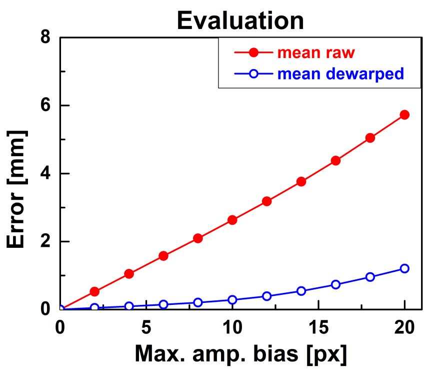


(a) Raw projection.

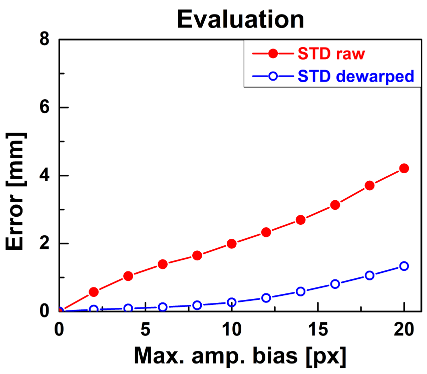


(b) Dewarped projection.

Figure 5.11: Examples of simulated evaluation results before (a) and after (b) distortion correction using warping maps. GT: ground truth; Proj.: projected; px: pixel.



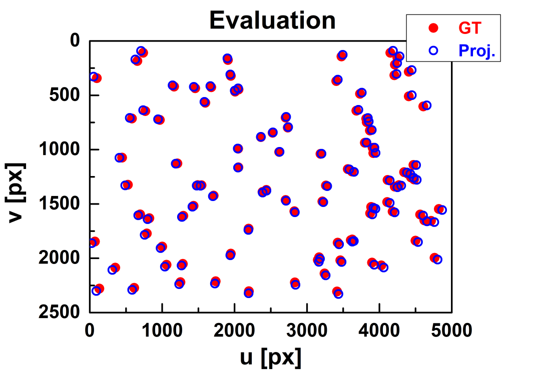
(a) Average projection errors before and after distortion correction.



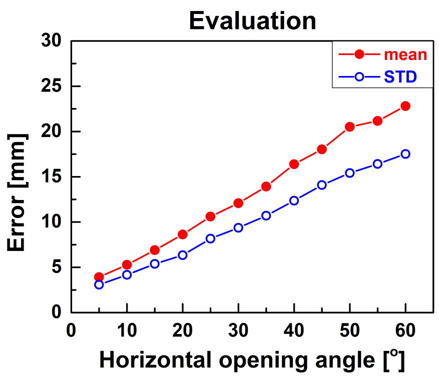
(b) STD of projection errors before and after distortion correction.

Figure 5.12: Statistics of evaluation results with various magnitudes of optical distortion. Max. amp.: maximal amplitude; STD: standard deviation; px: pixel.

is narrow, the calibration is insensitive to the viewpoint's displacement along the X^W–axis. Figure 5.13 (b) explicitly draws this conclusion. Both the average projection error in the evaluation phase and its standard deviation increase with the size of HUD-FOV. Therefore, for the calibration of AR-HUDs with larger opening angles, we have to be careful in selecting the eye box. For example, it is best to define a cube or sphere and use three linear axes aligned with the X^W–, Y^W– and Z^W–directions to move the calibration camera, respectively.



(a) An example evaluation with a tolerance of viewpoint displacement on the X^W–axis with $\sigma_{X^W} = 100\,\text{mm}$ under the horizontal opening angle of 60°.



(b) Statistics of projection with a tolerance of viewpoint displacement on the X^W–axis with $\sigma_{X^W} = 100\,\text{mm}$ under a series of horizontal opening angles.

Figure 5.13: Simulation results on various opening angles under the same angular resolution of the virtual image. GT: ground truth; Proj.: projected; px: pixel; STD: standard deviation.

5.2.8 Simulation with Chessboards

Since we employ two separated chessboard targets under the scheme in Section 4.5.1, we also simulate the scenario here. As is shown in Figure 5.14, we place a chessboard target with a 6×11 corner grid at two distances from the viewpoint. We repeat the simulation in Section 5.2.3.1, i.e. with 2D detection error. Other simulation schemes in Section 5.2.1–5.2.5 can also be repeated, though we do not show them concerning the conciseness of text. Figure 5.15 presents the evaluation examples where the separation between the boards is 1.5 m, which are similar to Figure 5.5.

When we use the chessboard target at two distances, we are also curious about the influence of different separations in between. The reason is that when the separation is not large enough, the practical experiment can degenerate to a semi-coplanar case, which is not supported by those PnP algorithms. The simulated results are plotted in Figure 5.16. We notice that both the average projection errors and their standard deviations are extremely unstable when the separation $d_{\text{sep}} < 2\,\text{m}$. This means we have to be careful that the two chessboard positions shall be chosen with a sufficient distance in between. Of course, the minimal requirement of such a separation may also depend on the precision of 3D detection, the number of control points, and other mentioned error sources. That is why

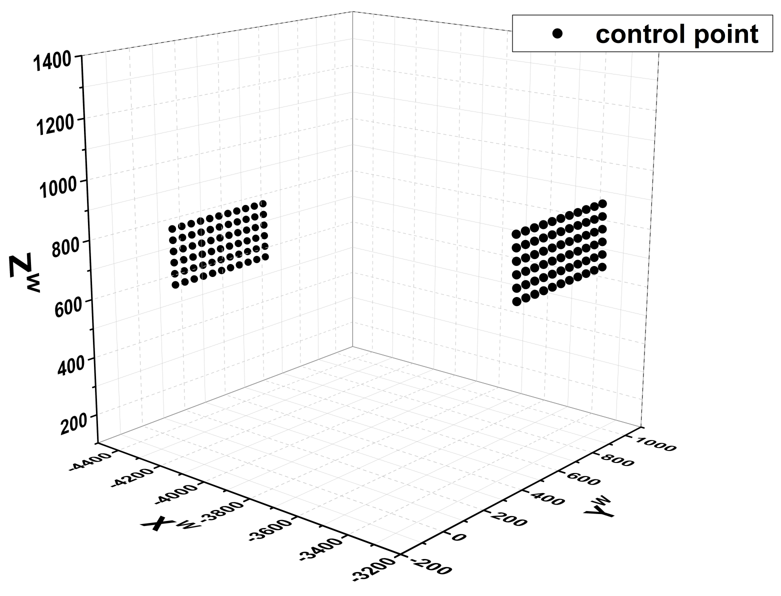


Figure 5.14: A simulated grid of 6 × 11 control points on a chessboard at two distances in the world space W.

we have adopted only a separation of around 0.8 m in the practical calibration scheme (see Section 4.5.1.1).

5.3 Pre-calibration of Cameras

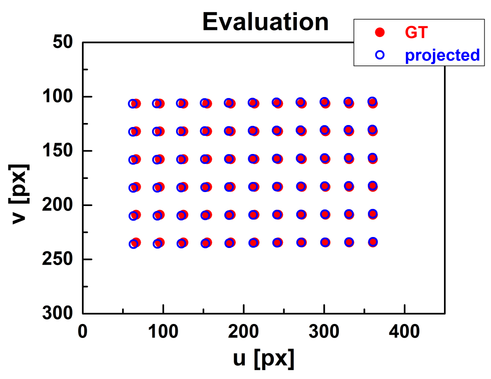
The pre-calibration results for the involving smartphone cameras, i.e. the Huawei P10 Lite and the Apple iPhone 7 Plus, are shown in Table 5.2 and 5.3, respectively. Table 5.4 and 5.5 present the calibrated parameters of the first ZED stereo camera used as the 3D sensor and the second ZED stereo camera applied as the driver camera.

Table 5.2: Calibration results of the smartphone camera Huawei P10 Lite. f_x and f_y are the horizontal and vertical focal lengths in pixel, respectively. $[x_0, y_0]^T$ is the principal point. k_1 and k_2 are the 1st and 2nd order radial distortion coefficients.

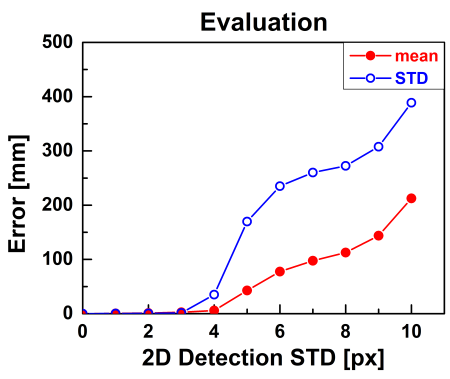
Camera type	**Huawei P10 Lite Monocular Camera**		
Resolution (px×px)	1920 × 1080		
f_x (px)	1517	f_y (px)	1502
x_0 (px)	964	y_0 (px)	509
k_1	0.185	k_2	-1.006

5.4 Manual Implementation Results (with Human Eyes)

Our manual AR-HUD calibration results correspond to the experimental approach in Section 4.4.1.1. The implementation was initially trialed in the laboratory environment using a home-made calibration target and human eyes, and then transplanted into some vehicles with supporting diagnostic interfaces. Here we show the test results on a Mercedes-Benz

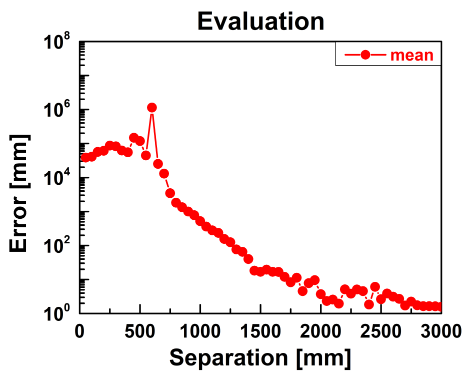


(a) Projection result when $\sigma_{uv} = 10.0$ px.

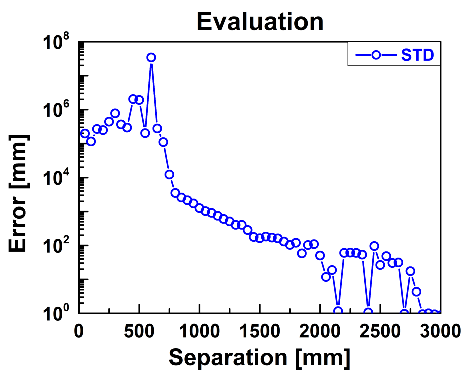


(b) Statistics on projection errors for a series of σ_{uv}.

Figure 5.15: Simulated evaluation results after calibration using two separated chessboards with random 2D detection errors. Both the two boards contain a 6 × 11 control point grid and are 1.5 m separated. Another chessboard with the same geometry is placed at an even farther distance for evaluation. GT: ground truth; STD: standard deviation; px: pixel; σ_{uv}: standard deviation of a single set of 2D detection errors.



(a) Average projection errors when $\sigma_{uv} = 10.0$ px.



(b) STD of projection errors when $\sigma_{uv} = 10.0$ px.

Figure 5.16: Statistics on simulated evaluations with two chessboard targets under different separations. We introduced a 2D detection error with a standard deviation $\sigma_{uv} = 3$ px. STD: standard deviation; px: pixel.

Table 5.3: Calibration results of the smartphone camera Apple iPhone 7 Plus. f_x and f_y are the horizontal and vertical focal lengths in pixel, respectively. $[x_0, y_0]^T$ is the principal point. k_1 and k_2 are the 1st and 2nd order radial distortion coefficients.

Camera type	**Apple iPhone 7 Plus Monocular Camera**		
Resolution (px×px)	1920 × 1080		
f_x (px)	1551	f_y (px)	1550
x_0 (px)	963	y_0 (px)	533
k_1	0.246	k_2	-1.440

Table 5.4: Calibration results of the 1st ZED stereo camera, which is used as the 3D sensor. f_x and f_y are the horizontal and vertical focal lengths in pixel, respectively. $[c_x, c_y]^T$ is the principal point. k_1 and k_2 are the 1st and 2nd order radial distortion coefficients.

Camera type	**ZED left eye**	**ZED right eye**
Resolution (px×px)	1920 × 1080	
baseline (mm)	120	
f_x (px)	1399	1401
f_y (px)	1398	1402
c_x (px)	991	1043
c_y (px)	601	548
k_1 (px)	-0.168	-0.168
k_2 (px)	0.026	0.025

Table 5.5: Calibration results of the 2nd ZED stereo camera, which is used as the driver camera. f_x and f_y are the horizontal and vertical focal lengths in pixel, respectively. $[c_x, c_y]^T$ is the principal point. k_1 and k_2 are the 1st and 2nd order radial distortion coefficients.

Camera type	**ZED left eye**	**ZED right eye**
Resolution (px×px)	1920 × 1080	
baseline (mm)	120	
f_x (px)	1401	1400
f_y (px)	1401	1400
c_x (px)	945	942
c_y (px)	587	580
k_1 (px)	-0.166	-0.169
k_2 (px)	0.025	0.026

S 500 car equipped with an AR-HUD system, which we parked in a workshop. As is declared before, using too many viewpoints or 2D–3D correspondence is impractical for the manual calibration. Hence, we defined an eye box containing only 3×3 train viewpoints (Figure 5.17) and mark 3×3 control points on a calibration target board. We did not include test viewpoints considering limited storage in the vehicle's control units and extra required human labor. The projection results in the evaluation phase at 7.5 m distance are plotted in Figure 5.18. Since there are not many point correspondences, we have not corrected the optical distortion because any distortion model can be underfitted in this case. Even so, we have observed that the projected 9 virtual points are tightly distributed to the measured ones at most of the train viewpoints. Here we show the situation of a single viewpoint, i.e. Viewpoint 3, as an example. Such projection errors are comparable to those under automatic implementations in the previous work [10], [24], [27].

At some other viewpoint, e.g. Viewpoint 7, the case becomes yet worse with a projection error larger than 10 mm. Several reasons can result in such instability, e.g. the 3D sensor or driver camera detection errors and the optical distortion. However, compared to the automatic calibration routines, a distinct error source is the operator's unfixed eye position. Though we have set a tolerance of 5 mm for the driver camera to verify the operator's view position, the head motion range of 1 cm can already affect the calibration accuracy. This effect has been demonstrated with the simulation in Section 5.2.5. We also notice that along with viewpoints on each horizontal line in the eye box (e.g. Viewpoint 1 to 3, 4 to 6, and 7 to 9), the average projection error decreases. This is mainly due to the changing display area on the windshield with varying curvature along Y^W–axis: since the driver's seat is on the left, if the viewpoint goes right, the optical path from the HUD projector goes right as well, and the reflection area for images on the windshield becomes flatter than on the left.

We have repeated such a manual implementation several times in the workshop with the car. Empirically for a rough AR-HUD calibration without using a calibration camera or much data to process, this manual manner with human eyes can already fulfill the basic calibration requirements in terms of accuracy.

5.5 Automatic Implementation Results

Next, we will show the calibration results under various automatic schemes that correspond to Section 4.5. They verify the involving schemes and provide a deeper understanding of the related concepts. In contrast to the relatively strenuous manual implementation, the automatic ones are chosen as counterparts of those in state of the art [10], [15]–[18], followed by a detailed comparison at a later stage in Chapter 6.

All the results come from calibrations of off-vehicle AR-HUD setups in our laboratory environment. We have used two different "HUD projector - windshield" pairs: one for the calibration using a sizable chessboard target, the other for the calibration using a piece of patterned paper and the target-free calibration.

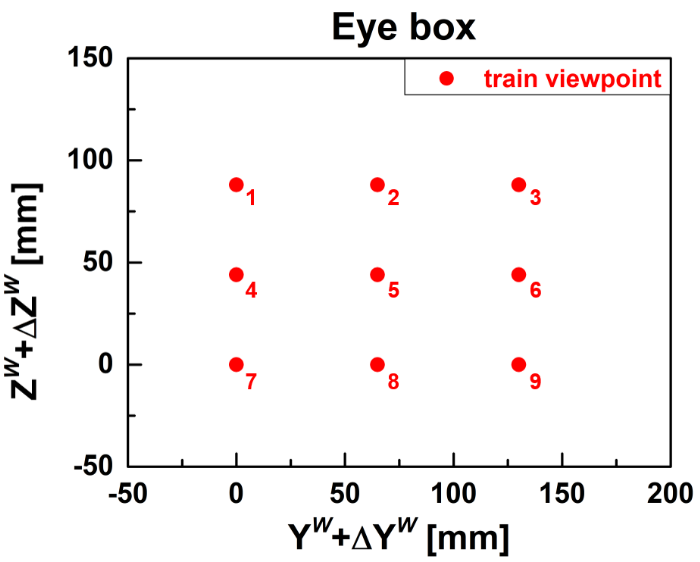
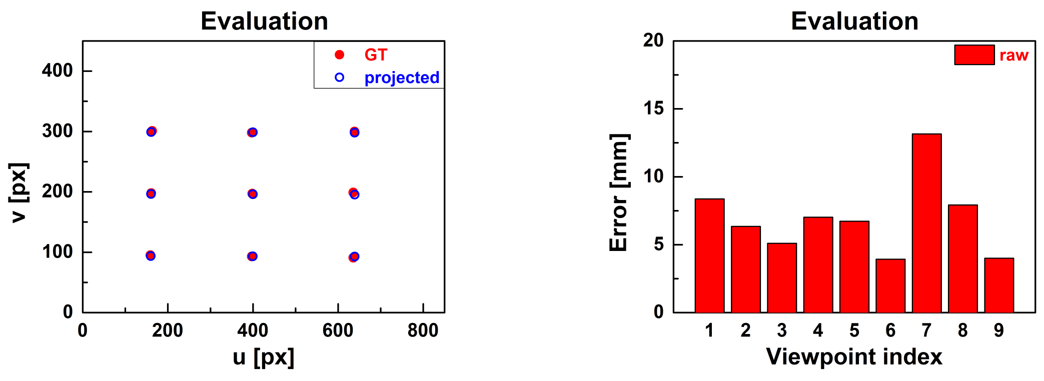


Figure 5.17: Selected 2D eye box with 9 train viewpoints on the $Y^W Z^W$–plane. ΔY^W and ΔZ^W represent the offsets on the corresponding axes.



(a) Evaluation at Viewpoint 3 using a 3×3 target board at 7.5 m.

(b) Statistics of evaluation results at all the train viewpoints in the 3×3 eye box.

Figure 5.18: Evaluation results of manual implementation using human eyes. There is no distortion correction because of too few applied control points on the target. GT: ground truth; px: pixel.

5.5.1 Calibration Results with a Chessboard Target

This scheme, as is described in Section 4.5.1, utilizes a conventional sizable chessboard pattern as the calibration target. At two different distances we finished the initial calibration and validation, and then at 7.5 m distance from the eye box, we accomplished the evaluation. Such a pattern has already been used for calibration of cameras [13].

5.5.1.1 Focal Lengths of HUD

We estimated the focal lengths f_u and f_v in pixel based on the approach in Section 4.5.1.2. Then we have $f_u = 4815$ pixel and $f_v = 5321$ pixel. The difference between these two focal lengths indicates the astigmatism in our AR-HUD.

5.5.1.2 Calibration Results at Train Viewpoints

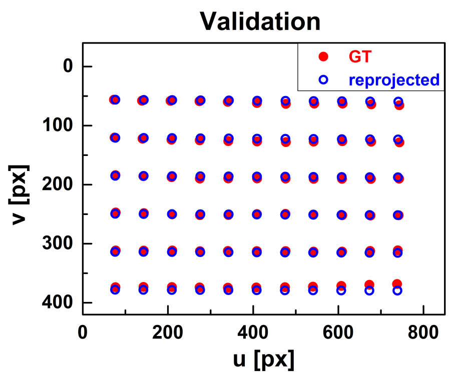
Figure 5.19 (a) shows an example of our AR-HUD calibration results at a train viewpoint. Using the acquired 2D–3D correspondences, we computed the extrinsic matrices $\mathbf{T}_{H_iW}$ for all the train viewpoints $V_{i,\text{tr}}$, respectively, followed by the validation step. The detected and reprojected virtual points are located close to each other with small biases. Regardless of measurement noises in the 3D detection and 2D pattern recognition, those residuals are mainly caused by optical distortion. Then for both u– and v– directions, we reconstructed the corresponding warping maps using the method described in Section 2.9.2, as is shown in Figure 5.20.

A raw evaluation result at the same train viewpoint is plotted in Figure 5.19 (b), where the projected virtual points are also well overlapping with the measured ground truth. From the above obtained warping maps, we selected the discrete bias vectors $[\Delta u_{ij}, \Delta v_{ij}]^T$ corresponding to all the projected $[u_{ij}, v_{ij}]^T$. We then dewarp the projections by adding these bias vectors, as shown in Figure 5.21 (a). Qualitatively we observe a closer overlap between measured and projected virtual points than in the raw evaluation. Till this step, we have accomplished initial calibration, validation and evaluation at train viewpoints $V_{i,\text{tr}}$. Figure 5.23 (a) collects the projection errors as statistics. According to the calculation in Section 5.1, the averaged projection error RMSE_va across all train viewpoints is 2.4 mm (1.1 arcmin), which is smaller than in state of the art. (Curve (c–d) in Fig. 9 of [10] shows approximately 3.8 mm i.e. 1.7 arcmin.)

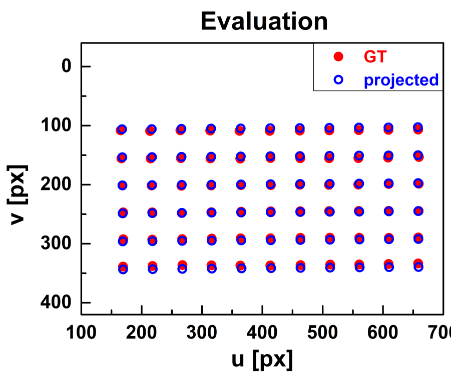
5.5.1.3 Driver Camera Calibration Results

Applying Eq. (4.1), we have acquired 3D information of each train viewpoint $V_{i,\text{tr}}$ in the world space W. Note that the X^W–component in the world coordinates was separately measured with a laser range finder (Leica DISTO™ D110), since in Section 3.2 and 5.2.7 we have concluded that this value could not be accurately acquired from calibration due to the HUD's narrow opening angles. Alternative measuring tools, such as a tape measure or ruler, should also be suitable.

Besides, we also have the viewpoint positions in the driver camera space D. Therefore, we computed the linear transformation $\mathbf{T}_{WD}$ from the driver camera to the world space

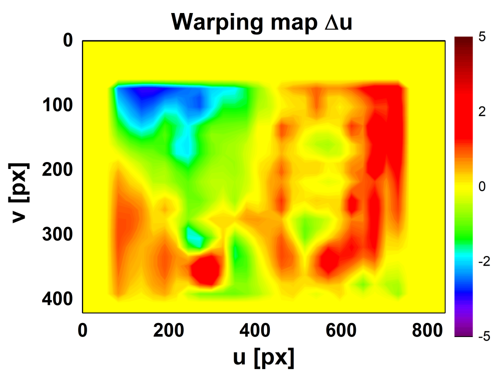


(a) Reprojection on the nearer target.

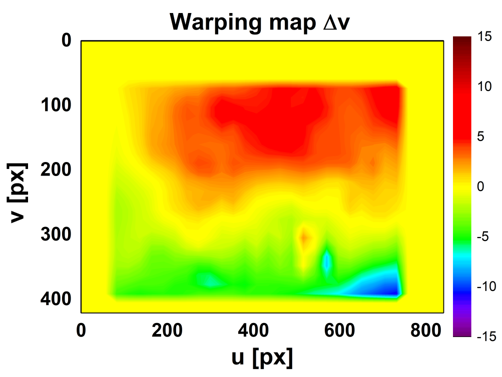


(b) Raw evaluation result at 7.5 m distance.

Figure 5.19: Validation and raw evaluation after calibrating the extrinsic matrix $\mathbf{T}_{H_iW}$ at Train Viewpoint 8 in Figure 4.8 using a 6×11 chessboard pattern as the target. GT: ground truth; px: pixel.



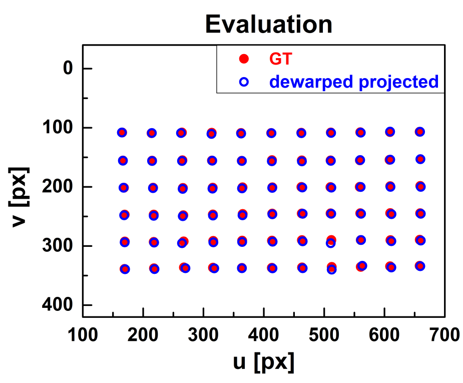
(a) Reconstructed warping map of Δu.



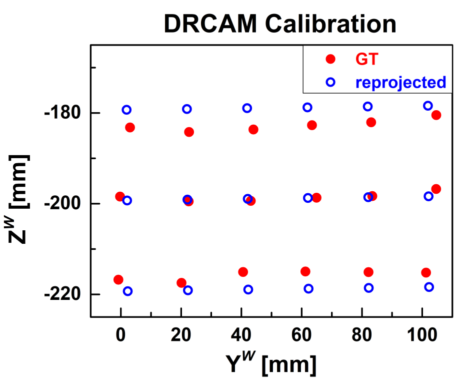
(b) Reconstructed warping map of Δv.

Figure 5.20: Warping maps corresponding to Train Viewpoint 8, which are reconstructed from discrete biases as continuous 2D fields. The yellow margins indicate the virtual image periphery that is not covered by the target due to its limited size, where the information of warping is actually missing.

and then reprojected the train viewpoints for comparison, as is shown in Figure 5.21 (b). The RMSE in Y^W– and Z^W–axis is 1.9 mm and 2.6 mm, respectively. Considering the size of eye box (40 mm × 100 mm), the reprojection errors are 1.9% and 6.5%.



(a) Dewarped projection, where porjected virtual points and ground truth are closer located than in Figure 5.19 (b).

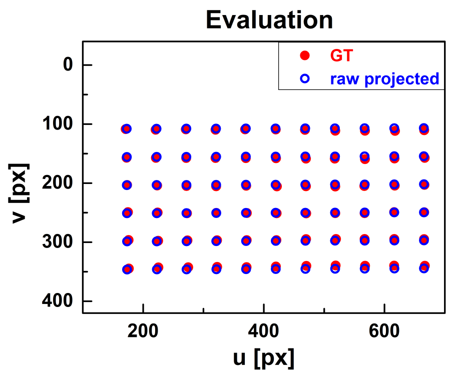


(b) Validation of driver camera calibration by reprojecting viewpoints to the world space W using the solved $\mathbf{T}_{WD}$. ΔY^W and ΔZ^W represent offsets.

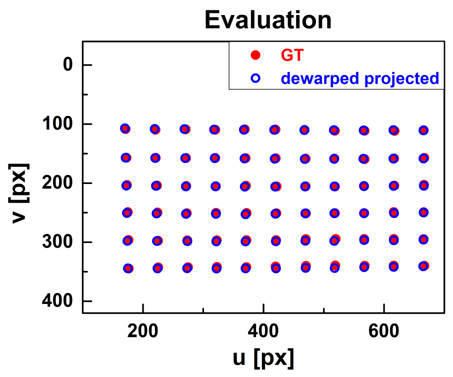
Figure 5.21: Dewarped evaluation result at Train Viewpoint 8 in Figure 4.8 and the driver camera calibration result. GT: ground truth; px: pixel.

5.5.1.4 Interpolation Results

The interpolation concept in Section 4.5.1.5 was implemented at the 10 test viewpoints in Figure 4.8, which includes interpolations of extrinsic parameters as well as warping maps. Figure 5.22 shows the projection results at a test viewpoint before and after distortion correction. Qualitatively we observe again that the projected virtual points are close to the measured ones, especially after dewarping. To further confirm the effectiveness of our distortion compensation, we show statistics on projection errors without/with distortion correction in Figure 5.23 (b). The RMSE$_{\text{te}}$ across all the test viewpoints is 2.5 mm (1.1 arcmin), which is also comparable to the result in [10] (Curve (a–b) in Fig. 9).

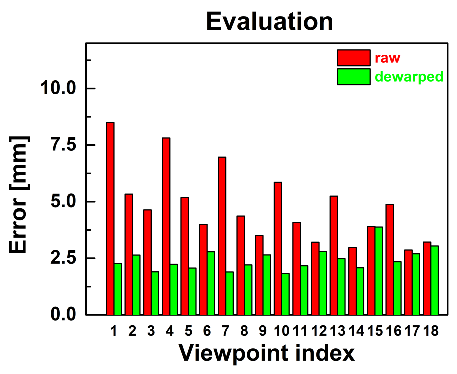


(a) Raw evaluation result at 7.5 m distance.

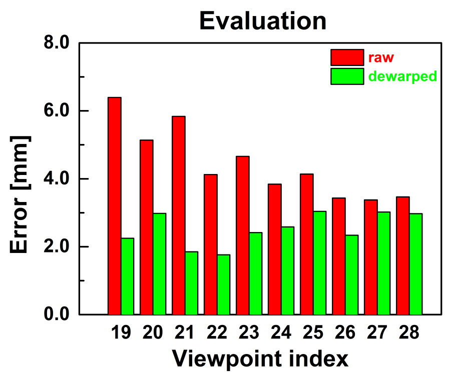


(b) Dewarped evaluation using interpolated warping maps.

Figure 5.22: Evaluation results at Test Viewpoint 24 in Figure 4.8 after interpolation of extrinsic matrices.



(a) Statistics of reprojection errors in evaluation at train viewpoints.



(b) Statistics of reprojection errors in evaluation at testing viewpoints.

Figure 5.23: Statistics of reprojection errors in the raw and dewarped evaluations. The viewpoint indices correspond to those in Figure 4.8.

5.5.2 Calibration Results with Patterned Paper

This scheme, as is described in Section 4.5.2, utilizes a piece of patterned paper as the calibration target for the initial calibration and validation phases, which can be laid on the outer surface of the windshield. In the evaluation phase, however, we employ a sizable chessboard target at 7.5 m from the eye box. This rectangular target paper has a size of 997 mm × 741 mm. A 12 × 7 chessboard with an 11 × 6 corner grid of corners is printed onside this target.

5.5.2.1 Focal Lengths of HUD

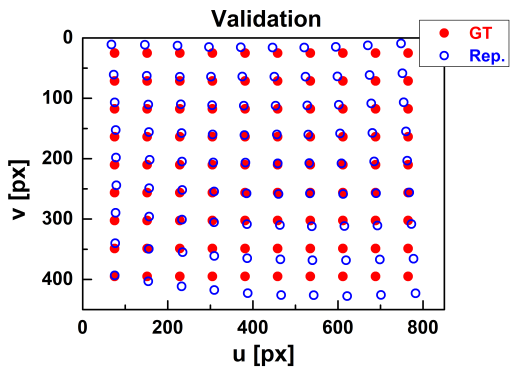
We estimate the focal lengths f_u and f_v in pixel based on the approach in Section 4.5.2.2. Then we have $f_u = 4242$ pixel and $f_v = 4089$ pixel. The difference between these two focal lengths indicates the astigmatism in our AR-HUD.

5.5.2.2 Extracted Warping Maps

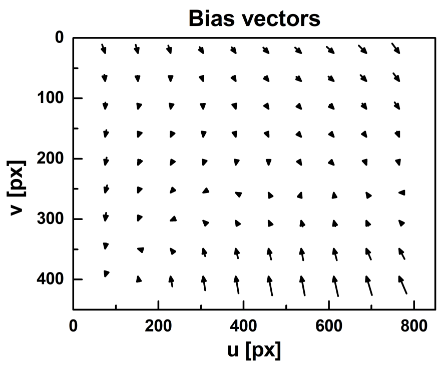
We extract warping maps through reprojection in the validation phase. Feeding the projection matrix $\mathbf{P}_{I_iV_i}$ into Eq. (2.5), we project all 3D reflection points Pr_{ij} in the view $V_{i,\mathrm{tr}}$ onto the virtual image I_i. An example at a train viewpoint is plotted in Figure 5.24 (a). Because of the nonlinear distortion, pixel errors exist between the ground truth input virtual points and reprojected ones. Therefore, we obtain the bias vectors in Figure 5.24 (b). Using bicubic interpolation, we reconstruct the warping maps for both Δu– and Δv–components, as are shown in Figure 5.25. Warping maps at other non-participating viewpoints should be obtainable through interpolation.

5.5.2.3 Qualitative and Quantitative Evaluation

As is stated in Section 1.2, a calibration routine for AR-HUDs is robust if and only if virtual objects align accurately with the real world at any feasible viewpoint V_i, including those uninvolved in the calibration. To examine our method's capability, we finally evaluate it at the 16 test viewpoints $V_{i,\mathrm{te}}$ using a chessboard target 7.5 m away from the eye box. We

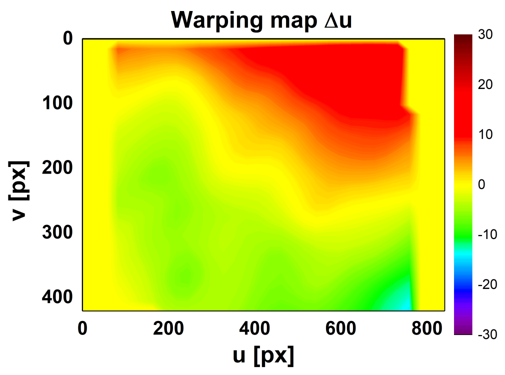


(a) Input and reprojected virtual points

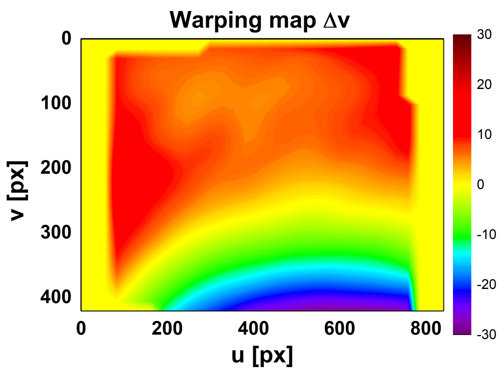


(b) Bias vectors pointing from reprojection to input

Figure 5.24: Reprojection and 2D biases in the virtual image, corresponding to Train Viewpoint 5 in Figure 3.2 (a). GT: ground truth; Rep.: reprojected; px: pixel.



(a) Warping map $w_{i,\Delta u}$

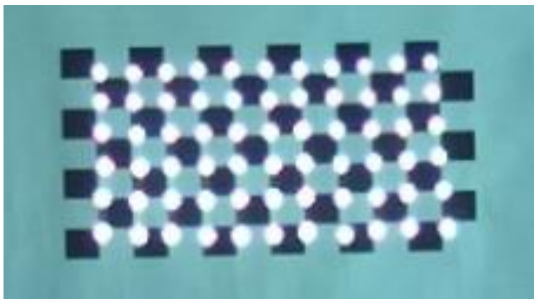


(b) Warping map $w_{i,\Delta v}$

Figure 5.25: Reconstructed warping maps for Train Viewpoint 5 in Figure 3.2 (a).

interpolate the respective extrinsic matrices $\mathbf{T}_{H_iW}$ and warping maps bilinearly using the calibrated ones at the 9 train viewpoints.

Our 3D sensor, i.e. the ZED stereo camera, detects the chessboard corners on the evaluation target. Then we managed to augment these corners with projected virtual points. Using the estimated intrinsic matrix $\mathbf{K}$ and interpolated extrinsic matrix $\mathbf{T}_{H_iW}$, we obtained these virtual points and rendered them via our AR-HUD. The picture captured by the smartphone camera is shown in Figure 5.26. Note that in the raw evaluation without distortion correction (Figure 5.26 (a)), the virtual points are already roughly aligned with the chessboard corners. After dewarping the virtual points using the interpolated warping maps, we realize a more accurate overlap (Figure 5.26 (b)).


(a) Raw evaluation

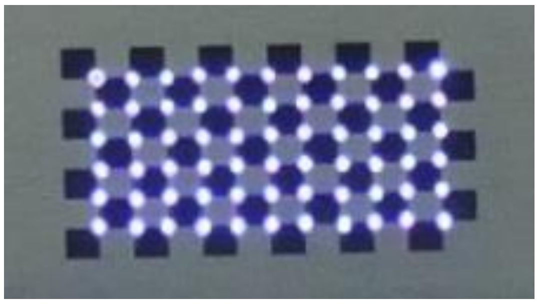

(b) Dewarped evaluation

Figure 5.26: Raw (a) and dewarped (b) qualitative evaluation results using a sizable chessboard target at 7.5 m. The pictures are captured by our smartphone camera at Test Viewpoint 16 in Figure 3.2 (a).

As for the quantitative evaluation, we search for centroids of projected virtual points in the above photos and compare them with the captured chessboard corners. Using Eq. (5.3) and (5.4), we can obtain the final RMSE across all the test viewpoints. Table 5.6 shows the statistics of such projection errors. The results are comparable with those in previous work [10], [27], which confirms that our AR-HUD is robustly calibrated.

Table 5.6: Statistics on projection errors in the evaluation at the test viewpoints under the calibration scheme using a piece of patterned paper.

Evaluation phase	RMSE_{te} (mm)	Angular accuracy (arcmin)	Standard deviation (mm)
Raw	13.3	6.1	6.0
Dewarped	4.6	2.1	2.7

5.5.3 Target-free Calibration Results

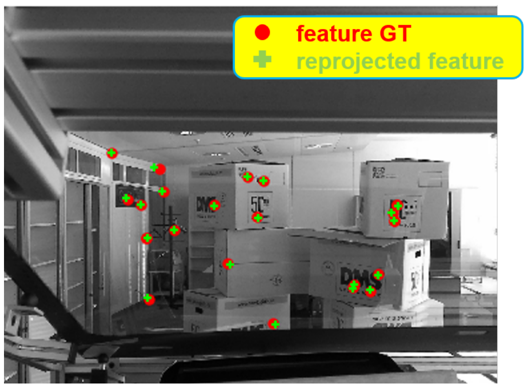
This scheme, as is described in Section 4.5.3, accomplishes the calibration of an AR-HUD in a target-free manner. We still employ the same chessboard target for the evaluation phase as in Section 5.5.2.

5.5.3.1 Focal Lengths of HUD

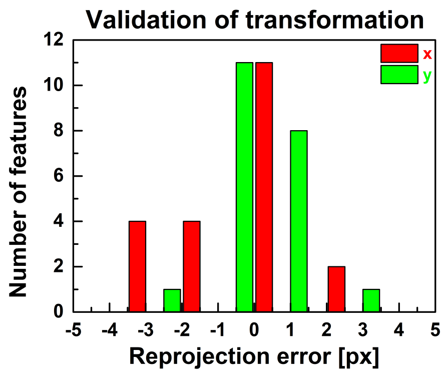
We adopt the method in Section 4.5.3.2 and estimate the focal lengths of our HUD in pixel. Then we have $f_u = 4574$ pixel and $f_v = 4054$ pixel as the horizontal and vertical focal lengths, respectively. The values are different than those in Section 5.5.2.1 because of changed view and random measurement errors.

5.5.3.2 Transformation from World to Viewpoints

As is described in Section 4.5.3, the transformation $\mathbf{T}_{V_iW}$ from the world W to a train viewpoint space $V_{i,\mathrm{tr}}$ is acquired using the frontal scene. Hence, we triangulate the inlier feature point pairs in the stereo images with our ZED stereo camera's pre-calibrated parameters (Table 5.4–5.5) to restore the 3D positions of the feature points P_j. With the calibrated matrix $\mathbf{T}_{V_iW}$ and the pre-calibrated smartphone camera's intrinsic matrix $\mathbf{K}_C$, we reproject these P_j onto the camera image whose resolution is Full HD, as is shown in Figure 5.27 (a). From the picture, we compare the ground truth and reprojection of those features by collecting the biases between them, as is plotted in Figure 5.27 (b). The mean value of the reprojection error is -0.4 pixel in the x–axis and 0.1 pixel in the y–axis. The standard deviation is 1.7 pixel and 1.1 pixel, respectively.



(a) Ground truth and reprojected features

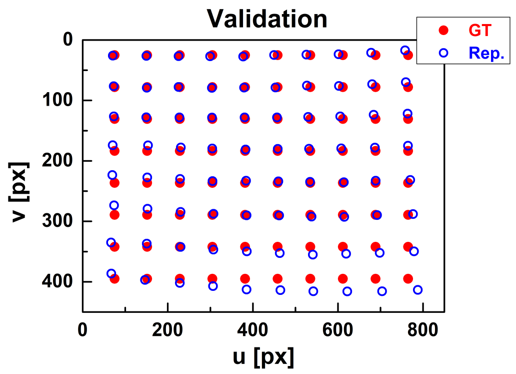


(b) Histogram of reprojection errors

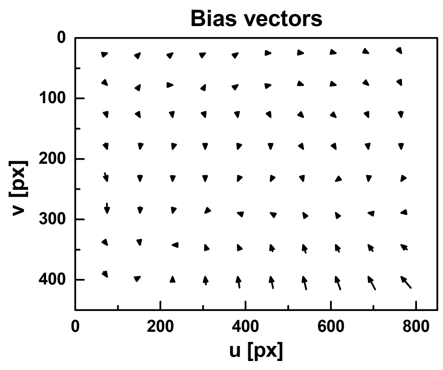
Figure 5.27: 21 inlier SIFT-based feature points and their reprojections from the 3D sensor to the calibration camera (a). We also plot statistics of the reprojection errors in pixel (b). GT: ground truth; px: pixel.

5.5.3.3 Validation and Warping Maps

As is stated in Section 4.5.3, the 3D control points are selected on the reconstructed optical rays $\mathbf{r}_{ij,\mathrm{tr}}$ corresponding to the train viewpoint $V_{ij,\mathrm{tr}}$, while the 2D virtual points $p_{j,\mathrm{tr}}$ are known as the array input. With these 2D–3D correspondences, we solve the rotation $\mathbf{R}_{H_iV_i}$ using the PnP algorithms. Applying it with the estimated HUD's intrinsic matrix $\mathbf{K}$, we obtain the reprojection results and the corresponding bias vectors. An example at Train Viewpoint 5 in the eye box (Figure 3.2 (a)) is plotted in Figure 5.28. We can notice that the optical distortion affects more on the edges or corners of HUD-FOV than in the central region. Using these bias vectors, we reconstruct the warping maps $w_{i,\Delta u}$ and $w_{i,\Delta v}$, as are shown in Figure 5.29.

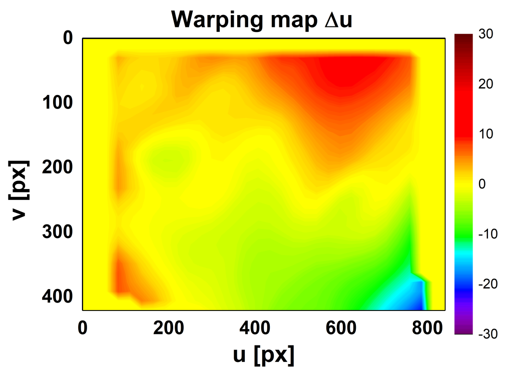


(a) Input and reprojected virtual points

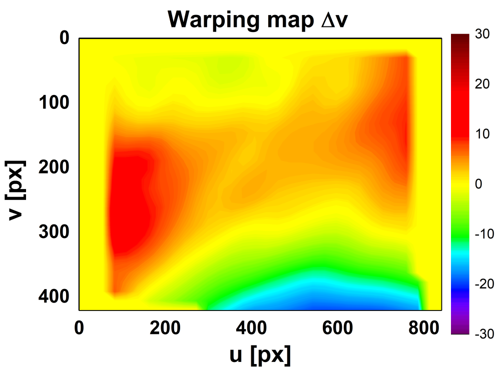


(b) Bias vectors pointing from reprojection to input

Figure 5.28: Reprojection for the validation phase and its corresponding 2D biases on the virtual image. Here we take Train Viewpoint 5 in the eye box (Figure 3.2 (a)) as the example. Rep.: reprojected; px: pixel.



(a) Warping map $w_{i,\Delta u}$

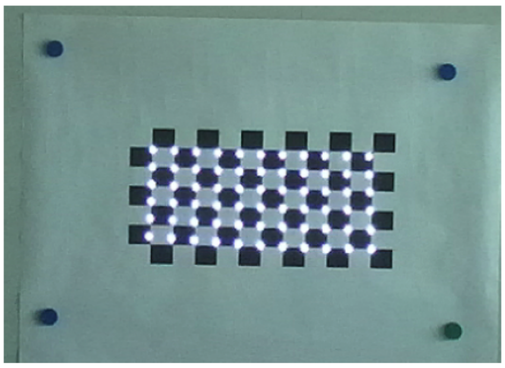


(b) Warping map $w_{i,\Delta v}$

Figure 5.29: Reconstructed warping maps for Train Viewpoint 5 in Figure 3.2 (a). Here the peripheries are also reconstructed by extrapolation.

5.5.3.4 Evaluation and Interpolation Results

The qualitative evaluation approach is the same as in Section 5.5.2.3. We first acquire the transformation $\mathbf{T}_{V_iW}$ and rotation matrices $\mathbf{R}_{H_iV_i}$ for all the test viewpoints $V_{i,\text{te}}$ via interpolation. Then, the 3D sensor detects the chessboard corners' 3D positions. For each test viewpoint, feeding the estimated matrices $\mathbf{K}$, $\mathbf{T}_{V_iW}$ and $\mathbf{R}_{H_iV_i}$, we can project the 3D positions onto the virtual image I_i. Finally, we render the obtained virtual points, and observe the alignment between them and chessboard corners with the smartphone camera. The qualitative results are demonstrated in Figure 5.30. We can see that the projection after distortion correction is more accurate than before it.


(a) Raw evaluation

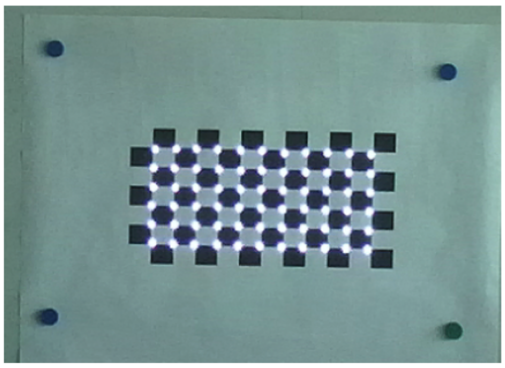

(b) Dewarped evaluation

Figure 5.30: Qualitative evaluation results of the target-free calibration scheme. The raw projected virtual points align close with the chessboard corners (a). After distortion correction using the reconstructed warping maps, we acquire a more precise alignment (b).

As quantitative results, we make statistics about reprojection errors and angular accuracy, which is listed in Table 5.7.

Table 5.7: Statistics on reprojection errors for the test viewpoints $V_{i,\text{te}}$ in the evaluation phase.

Evaluation phase	RMSE_{te} (mm)	Angular accuracy (arcmin)	Standard deviation (mm)
Raw	11.5	5.0	8.4
Dewarped	6.7	3.1	4.7

5.5.4 Summary

We note that all of our automatic calibration pipelines lead to tiny projection errors in the evaluation phase for test viewpoints. Till now, they have been validated as effective AR-HUD calibration methods. Remarkably, the method using a chessboard target has achieved a better accuracy in the evaluation phase than the previous work [10]. However, these schemes have respective advantages and disadvantages. A detailed comparison will be done in Section 6.1.

6. Discussion

This chapter opens a discussion on several aspects of the AR-HUD calibration, which is highly related to our proposed concepts. We first compare our schemes with state of the art in Section 6.1. The comparison involves accuracy, time-efficiency, and target complexity. Section 6.2 focuses on the virtual image plane, where we reveal its changing behavior at multiple viewpoints, though in each approach in Section 4.5, we have estimated the HUD's intrinsic parameters only once. Next, we trial the DLT algorithm in Section 6.3 and explain why we did not apply it in the AR-HUD calibration. Finally in Section 6.4, we compare our distortion model, i.e. warping maps, with conventional camera lens distortion model and polynomial regression model.

6.1 Comparison among Schemes

In this section, we compare the automatic calibration schemes because, in the literature, they generally use automatic approaches with calibration cameras. We exclude manual implementation in case of the influence of artificial errors, such as the head motion or manual visual perception error. The discussion covers our proposed schemes [24], [27], [56], [57] and those from the previous work [10], [15]–[18]. We consider various aspects in terms of both quality and operability in the automotive industry.

6.1.1 Calibration Accuracy

We demonstrate all the projection errors available from our experiments and state of the art to compare the calibration accuracy. Note that for a fair comparison, we have always employed the target in the evaluation phase at 7.5 m from the eye box, which is in line with that in the previous work [10]. Some other literature [15]–[18] only demonstrates qualitative evaluations of their calibration methods, whereas their quantitative results are not available.

From Table 6.1 we observe that our automatic calibration scheme using a chessboard yields better angular accuracy than that using a piece of patterned paper or the target-free approach. This can be understood because a sizable calibration target is more favorable for 3D sensing so that the positions of control points are restored more accurately. The projection error is also smaller than that in state of the art. Nonetheless, considering that different HUD projectors and windshields were used and the values still fall in the same order of magnitude, we cannot curtly claim which approach performs the best. We can yet conclude that all these different schemes offer comparable, accurate results that can already fulfill the automotive industry's requirement.

6.1.2 Time Efficiency

The time consumption mainly includes two aspects: preparation for the experiments and the implementation duration. Regardless of the development and integration of hardware

Table 6.1: Average angular accuracy in the evaluation phases of various experimental schemes.

Scheme	**Wientapper et al. [10]**	**Using chessboard**	**Using patterned paper**	**Target-free**
Angular accuracy	1.7 arcmin	1.1 arcmin	2.1 arcmin	3.1 arcmin

devices or software, we emphasize the deployment of calibration targets and ad-hoc components. Some approaches require less labor in this process than others, while others may take less time when the calibration operation starts (or vice versa). Note that the latter depends on the number of involving train viewpoints.

Table 6.2 lists the time consumptions in our three automatic calibration schemes that correspond to Section 4.5.2, 4.5.1 and 4.5.3. Note that though the target-free scheme omits the calibration target's preparation, it does not mean that there is no preparation time because we have to select a proper frontal scene for our pattern recognition algorithms to extract enough feature points. Nevertheless, this scheme generally performs best in accelerating the implementation according to our experience. We can imagine that in the future if the workshop deploys a suitable frontal scene for cars, the target-free scheme will save much more time accumulatively as more vehicles equipped with AR-HUDs are calibrated.

Table 6.3 shows the comparison of time-consumption between our fastest scheme, i.e. the target-free one, and those in certain previous work [10], [18]. Other publications [15]–[17] did not disclose information on their time-efficiency. Moreover, we can only provide the available data because Deng et al. [18] did not prepare a target, while Wientapper et al. [10] did not publish their implementation duration at each viewpoint. Nevertheless, this comparison still implies that our target-free calibration scheme has shown a satisfying time-efficiency, indicating high applicability in the automotive industry.

Table 6.2: Time consumption of our proposed automatic AR-HUD calibration schemes.

Scheme	**Using chessboard**	**Using patterned paper**	**Target-free**
Target preparation	$\sim 10\,\text{min}$	$\sim 5\,\text{min}$	-
Running per viewpoint	$\sim 1\,\text{min}$	$\sim 1\,\text{min}$	$\sim 10\,\text{s}$

6.1.3 Target Complexity

Table 6.4 shows a comparison between our schemes and those in previous literature [10], [15]–[18] in terms of target complexity. Wientapper's method [10] requires the most ad-hoc components to support the experiment. Our scheme in Section 4.5.1 demands a sizable calibration target that can be properly placed in front of the windshield. Both the

Table 6.3: Comparison between our target-free AR-HUD calibration scheme and state of the art with regard to time consumption.

Scheme	**Our target-free**	**Wientapper et al. [10]**	**Deng et al. [18]**
Target preparation	-	30 min to 45 min	-
Running per viewpoint	$\sim 10\,\mathrm{s}$	-	$< 1\,\mathrm{min}$

above bring extra costs for the factory or workshops. In contrast, our other schemes in Section 4.5.2 and 4.5.3 require less efforts in preparing calibration targets. Hence, they are potential for the customers to calibrate their automotive AR-HUDs themselves once related user-interfaces are available.

Table 6.4: Target complexities in different AR-HUD calibration schemes, including those in state of the art.

Scheme	**Our chessboard-based**	**Our patterned-paper-based**	**Our target-free**	**Wientapper et al. [10]**
Target complexity	A sizable chessboard pattern	A piece of patterned paper	Fontal scene with features	Canvas, textured covers, markers
Scheme	**Hosseini et al. [15]**	**Ueno and Komuro [16]**	**Yoon and Kim [17]**	**Deng et al. [18]**
Target complexity	A chessboard pattern	Target-free, but relies on precise camera orienting	Fontal scene with features	Target-free, but relies on mixed reality glasses

6.2 View-dependent Virtual Images

In this section, we investigate the changing behavior of HUD's focal lengths with viewpoints. From the calibrated extrinsic matrix $\mathbf{T}_{H_iW}$, we can recover the Euler angles, i.e. roll (α), pitch (β) and yaw (γ) angles, and the translation vector $\mathbf{t}_{H_iW}$, as is stated in Section 2.2. Since we have also tracked the positions of all the train viewpoints $V_{i,\mathrm{tr}}$ in the world space W using the driver camera, we can readily reconstruct the chief rays $\mathbf{r}_{i,\mathrm{tr}}$ starting from the individual viewpoint and passing through the corresponding centers of virtual images I_i, as is mentioned in the caption of Figure 2.3.

We apply some measured data from the manual experimental scheme (see Section 4.4.1.1) and plot the reconstructed chief rays in Figure 6.1. Notice that these chief rays are not exactly intersecting with each other at a common 3D point. Regardless of measurement errors, a possible reason is the influence of optical distortion. Alternatively speaking, because of the curved windshield and reflective mirrors in the HUD projector, the residual distortion leads to a deformed virtual image plane whose pose and geometry become, in fact, view-dependent. Our analysis indicates that the assumption of a view-independent virtual image plane in state of the art [10], [18] is only a rough approximation. Therefore, throughout this thesis, we have not adopted this conventional premise.

Nonetheless, the view-dependent virtual images imply view-dependent intrinsic matrices $\mathbf{K}$ of the HUD. This has interpreted the difference between the acquired HUD's focal lengths in Section 5.5.2.1 and 5.5.3.1. Hence, we should have estimated the intrinsics at each train viewpoint $V_{i,\mathrm{tr}}$ separately instead of measuring it only once. However, repeating this estimation is unnecessary because the eye box size is much smaller than the HUD's projection distance, and the resulted biases can be readily mitigated by the warping maps in the distortion correction.

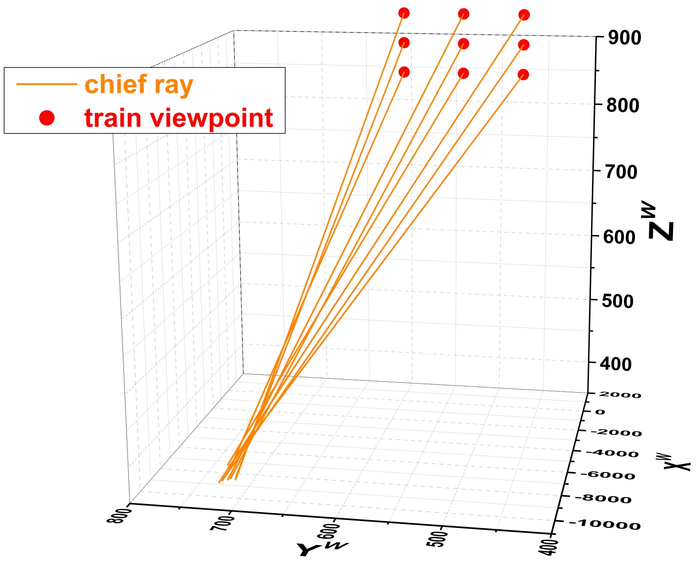


Figure 6.1: Reconstructed chief rays $\mathbf{r}_{i,\mathrm{tr}}$ starting from the individual viewpoint $V_{i,\mathrm{tr}}$ and passing through the corresponding centers of virtual images. We can observe that they are not exactly intersecting with each other in the world space W.

6.3 Direct Linear Transformation

Besides the PnP algorithms, we trial the DLT algorithm to accomplish the calibration. The main difference between them is: the former requires knowledge or pre-estimation about intrinsic matrix $\mathbf{K}$ to recover the extrinsic transformation $\mathbf{T}_{H_iW}$; on the contrary, the latter directly recovers the projection matrix $\mathbf{P}_{I_iW}$ followed by a decoupling procedure to offer $\mathbf{K}$ and $\mathbf{T}_{H_iW}$, respectively. Nevertheless, it is sometimes not easy to decouple this matrix precisely. Here we take the experimental data from the scheme in Section 5.5.1 as an example for the analysis of the DLT algorithm. Accordingly, the selected eye box is identical to Figure 4.8.

The projection matrix $\mathbf{P}_{I_iW}$ is a 3×4 matrix having the following form:

$$\mathbf{P}_{I_iW} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}, \tag{6.1}$$

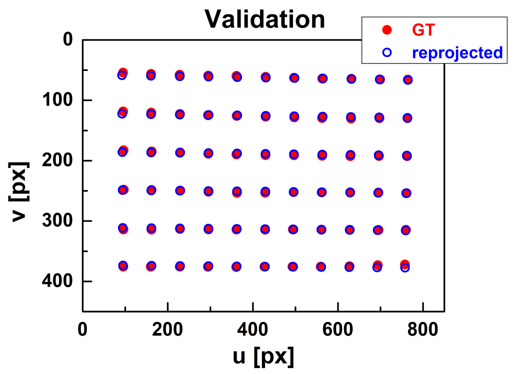
where m_{ij} are normalized in term of $m_{34} = 1$ for convenience. From Eq. (2.7)–(2.9) we have:

$$\mathbf{P}_{I_iW} = \mathbf{K}\mathbf{T}_{H_iW} = \mathbf{K}\begin{bmatrix}\mathbf{R}_{H_iW} & \mathbf{t}_{H_iW} \\ \mathbf{0} & 1\end{bmatrix} = \begin{bmatrix} f_u & s & u_0 & 0 \\ 0 & f_v & v_0 & 0 \\ 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1\end{bmatrix}. \tag{6.2}$$

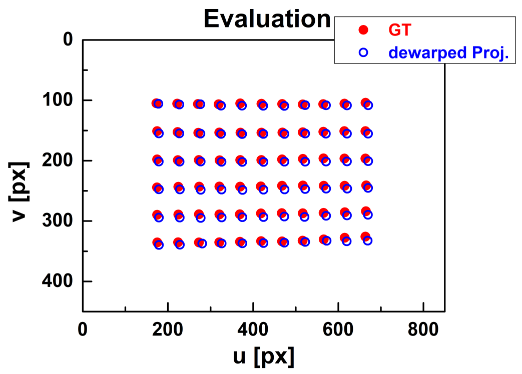
We show the restored projection matrix $\mathbf{P}_{I_{11}W}$ at the Train Viewpoint 11 here in Table 6.5. Figure 6.2 shows the validation result at Viewpoint 11, and the evaluation result at Test Viewpoint 23 after distortion correction. To acquire the projection matrix $\mathbf{P}_{I_{23}W}$, we directly interpolate its elements using those matrices obtained at neighboring train viewpoints, while the distortion correction is accomplished using correspondingly reconstructed warping maps. We observe that the reprojected virtual points in the validation phase (Figure 6.2 (a)) are located closer to the measured ground truth than the projected ones in the evaluation phase (Figure 6.2 (b)).

Table 6.5: Normalized calibrated projection matrix $\mathbf{P}_{I_iW}$ using the DLT algorithm at Viewpoint 11 under the scheme in Section 5.5.1.

m_{11}	-0.4919	m_{12}	20.11	m_{13}	-0.5817	m_{14}	-846.0
m_{21}	0.3756	m_{22}	-0.0725	m_{23}	-21.58	m_{24}	-4478
m_{31}	-0.0041	m_{32}	0.0004	m_{33}	-0.0011	m_{34}	1.000



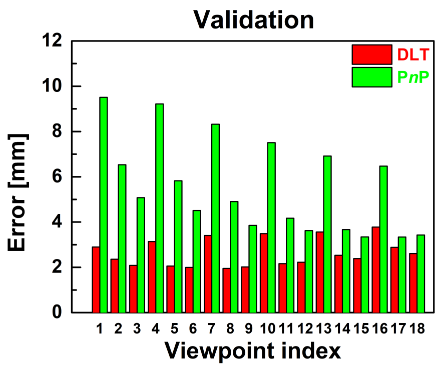
(a) Validation result at Viewpoint 11.



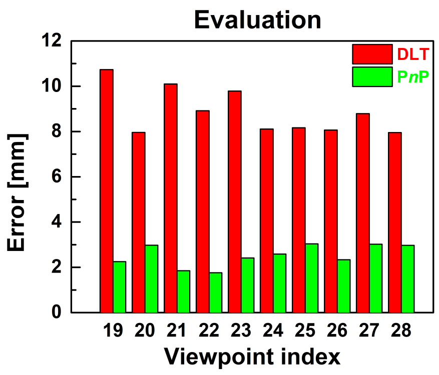
(b) Evaluation result at Viewpoint 23.

Figure 6.2: Validation (a) and evaluation (b) results using the DLT algorithm. The viewpoints are selected in the eye box shown in Figure 4.8. Note that we show the validation on the nearer target distance here. GT: ground truth; Proj.: projected; px: pixel.

Figure 6.3 shows the statistical comparison between the calibration results based on the DLT algorithm and that based on PnP algorithms. Both data sets are acquired from the automatic calibration scheme in Section 5.5.1. We notice that the validation results using the DLT algorithm are better than using PnP algorithm in terms of reprojection error, while the evaluation performances in terms of projection error are in contrast. The reason is that while calculating the projection matrix $\mathbf{P}_{I_iW}$, some mathematical constraints, such as the hidden orthonormality of the rotation matrix $\mathbf{R}_{H_iW}$, are missing. Therefore, the validation performs better, but the interpolation is more vulnerable to noises from the implementation or overfitted elements in $\mathbf{P}_{I_iW}$ at the train viewpoints. However, we accept that the DLT algorithm is also applicable for AR-HUD calibration if there is no requirement for a precise recovery of the AR-HUD system's intrinsic and extrinsic characteristics.



(a) Validation result at the train viewpoints.



(b) Dewarped evaluation result at the test viewpoints.

Figure 6.3: Comparison between the calibration results acquired using PnP and DLT algorithms at multiple viewpoints whose indices are in line with those in Figure 4.8.

Next, we introduce the decoupling of intrinsic ($\mathbf{K}$) and extrinsic ($\mathbf{T}_{H_iW}$) matrices from the projection matrix $\mathbf{P}_{I_iW}$ acquired from the DLT algorithm. We shall apply the orthonormality of the rotation matrix $\mathbf{R}_{H_iW}$ explicitly [75], while keeping $m_{34} = 1$. Then we have:

$$\begin{cases} t_3 = \|m_{3,1:3}\|^{-1} \\ u_0 = t_3^2 \cdot \langle m_{1,1:3}, m_{3,1:3} \rangle \\ v_0 = t_3^2 \cdot \langle m_{2,1:3}, m_{3,1:3} \rangle \\ f_u = t_3^2 \cdot \|m_{1,1:3} \times m_{3,1:3}\| \\ f_v = t_3^2 \cdot \|m_{2,1:3} \times m_{3,1:3}\| \\ t_1 = t_3(m_{14} - u_0) \cdot f_u^{-1} \\ t_2 = t_3(m_{24} - v_0) \cdot f_v^{-1} \\ r_{1,1:3} = t_3(m_{1,1:3} - u_0 m_{3,1:3}) \cdot f_u^{-1} \\ r_{2,1:3} = t_3(m_{2,1:3} - v_0 m_{3,1:3}) \cdot f_v^{-1} \\ r_{3,1:3} = t_3 m_{3,1:3} \end{cases}, \tag{6.3}$$

where $\langle , \rangle$ represents the inner product of two vectors, $\|\cdot\|$ represents the norm of a vector, $\times$ represents the cross product between two vectors, and $a_{i,j:k}$ denotes a row vector containing the sequential elements a_{ij}, ..., a_{ik} in the ith row of a matrix $\mathbf{A}$.

Table 6.6 lists the restored focal lengths from the DLT algorithm and those estimated for the PnP algorithm. As we see, the two groups differ obviously from each other. Moreover, suppose we interpolate the projection matrix $\mathbf{P}_{I_iW}$ directly for test viewpoints. In that case, we have to interpolate 11 elements separately (under the condition that $m_{34} = 1$). We cannot guarantee this naive interpolation's quality because the results might not obey mathematical consistency (e.g. orthonormality of the hidden rotation matrix $\mathbf{R}_{H_iW}$) under the influence of measurement errors. On the contrary, when using PnP algorithms, we only have to interpolate 6 terms, and they are mathematically consistent. Therefore, we insisted on first estimating the intrinsic matrix and then using PnP algorithms to calibrate the extrinsic ones. This has already provided us robustness in dealing with multiple viewpoints.

Table 6.6: Comparison between intrinsic parameters acquired from two approaches: the first group is estimated under stereo vision in Section 4.5.1.2, which we regard as ground truth (GT); the second group is decoupled from the projection matrix $\mathbf{P}_{I_{11}W}$ that we calibrated using the DLT algorithm (here we take the data at Viewpoint 11 in Figure 4.8 as example). px: pixel.

Quantity	f_u (px)	f_v (px)
Intrinsic parameters (GT)	4815	5321
Intrinsic parameters (DLT)	4689	4930
Relative Error (%)	-2.6	-7.3

6.4 Distortion Models & Comparison

In this section, we discuss different distortion models and compare them under our calibration circumstances. We focus mainly on the models that we have ever applied in our published work: the conventional camera distortion model [27] and bicubically interpolated warping maps [24], [56], [57], which are introduced in Section 2.9.1 and 2.9.2, respectively. The experimental data are taken from the scheme in Section 4.5.1. We also review the polynomial model in the previous literature [10], [18].

6.4.1 Camera Distortion Model

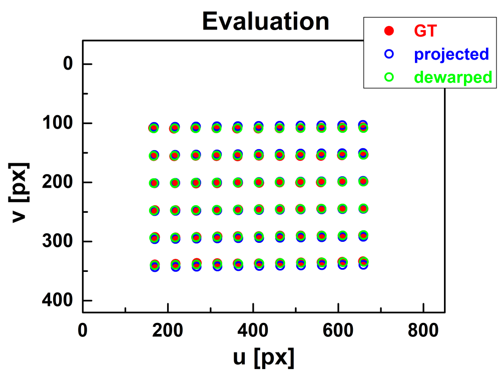
We take Eq. (2.39) as the standard form to describe the camera distortion model. Firstly, we consider every single train viewpoint $V_{i,\text{tr}}$. Because all the distortion coefficients (k_1, k_2, p_1, p_2, q_1 and q_2) are extracted from the validation phase, we have to examine their effectiveness in the evaluation phase. This means we rectify the raw projected virtual points using these coefficients and observe the correction outcome. We further consider those test viewpoints $V_{i,\text{te}}$ by acquiring their respective distortion coefficients from a bilinear interpolation using those of the train set and then substituting them again into the evaluation phase.

To qualitatively demonstrate our results, we select a viewpoint (Train Viewpoint 8) from the train set and another one (Test Viewpoint 24) from the test set (see also

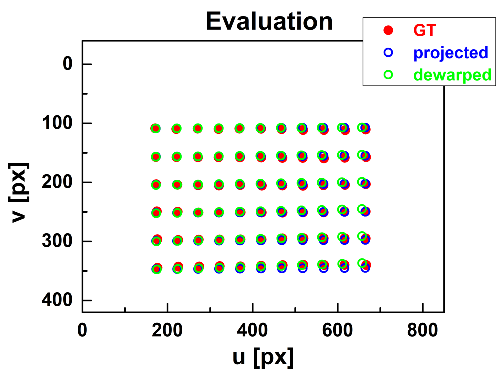
Figure 4.8). As are shown in 6.4, we plot the measured (ground truth), projected, and rectified projected virtual points together. We notice that at Train Viewpoint 8, the distortion is suppressed, while at Test Viewpoint 24, some dewarped projections are more separated from the ground truth than the corresponding raw projected ones, especially those on the right part of the virtual image. The possible reason is that for a fixed viewpoint, the optical distortion is stronger on the right part than other regions, which is not fully compensable using the interpolated distortion coefficients.

We also show statistics in Figure 6.5 on the projection errors before and after employing the camera distortion coefficients in the evaluation phase. For each train viewpoint, the distortion model helps reduce projection errors, but for each test viewpoint, the interpolated distortion coefficients show no such obvious rule. Nonetheless, at Test Viewpoint 19 and 22, we encounter outliers, where the distortion correction makes the projections even much worse.

In conclusion, though the conventional parametric camera distortion model can compensate the optical distortion individually for each train viewpoint, its effectiveness is doubtable for other non-participating viewpoints inside the eye box after interpolation. Therefore, to use this precisely, we have to repeat our implementation densely at many viewpoints. For example, suppose we have an 80 mm × 60 mm planar eye box, empirically we have to set the interval between neighboring viewpoints to 5 mm, that means there are total $17 \times 13 = 221$ viewpoints to calibrate. Even if the implementation at each of them takes 30 s, the initial calibration phase takes totally around 2 h, which is intolerable for automotive factories or workshops.



(a) Evaluation at Train Viewpoint 8.



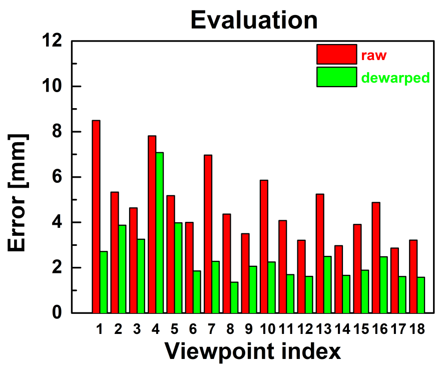
(b) Evaluation at Test Viewpoint 24.

Figure 6.4: Evaluation results at a train viewpoint and a test viewpoint based on the camera distortion model. GT: ground truth; px: pixel.

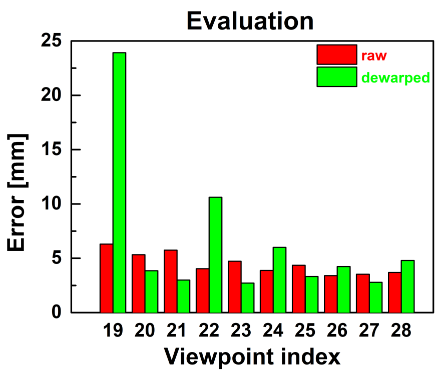
6.4.2 Warping Maps

The power of warping maps for distortion correction is already proven with the results in Section 5.5. Here we mainly discuss its advantages and disadvantages, especially compared to the conventional camera distortion model.

The warping map is a nonparametric method to correct the optical distortion. It is reconstructed from locally bias vectors via a bicubic interpolation. In contrast to the



(a) Train set.



(b) Test set.

Figure 6.5: Statistics on raw and dewarped projections from the evaluation phase at the train and test viewpoints, which are based on camera distortion model. All the distortion coefficients come from the validation at corresponding train viewpoints and are applied to test viewpoints via interpolation.

parametric camera distortion model, it recovers much more details in the virtual image I_i, even deep into each pixel. These details may also help understand the variation of distortion across the reflection area and examine the windshield's quality. Hence, our warping maps have the same resolution as the virtual image and occupy much more storage space than the parametric models. For example, if we have an 800×400 virtual image and an eye box including 25 train viewpoints, we should have to store a total of $800 \times 400 \times 25 = 8 \times 10^6$ bias vectors, i.e. a total of 1.6×10^7 numbers for the warping maps $w_{i,\Delta u}$ and $w_{i,\Delta v}$. Assuming the distortion is not extremely severe, and all these numbers fall within the interval $[-128, 127]$ (the range of a byte in the computer), then we need at least 15.3 MB storage space. In contrast, if we use the camera distortion model in Eq. (2.39), we only have 6 coefficients for each viewpoint. Suppose the coefficients are represented in float type (4 bytes for a float number), and the eye box remains the same, then we need only $6 \times 4 \times 25 = 0.6$ KB storage space, which is about 2.7×10^4 times smaller than using warping maps. However, in the laboratory environment, i.e. regardless of storage capacity, using warping maps is a more robust option. We can also declare that if the on-vehicle computer's storage and computational capacity fulfill the requirement, warping maps are highly recommendable for distortion correction.

6.4.3 Polynomial Regression Model

Some state of the art applied polynomial regression models to fit the optical distortion. Deng et al. [18] defined a regression model as:

$$\begin{cases} \hat{u}_{ij} = \sum_{d_1,d_2,d_3} b_{ij,u}\left(d_1,d_2,d_3\right)\left(\hat{X}^W_{V_i,\mathrm{tr}}\right)^{d_1}\left(\hat{Y}^W_{V_i,\mathrm{tr}}\right)^{d_2}\left(\hat{Z}^W_{V_i,\mathrm{tr}}\right)^{d_3} \\ \hat{v}_{ij} = \sum_{d_1,d_2,d_3} b_{ij,v}\left(d_1,d_2,d_3\right)\left(\hat{X}^W_{V_i,\mathrm{tr}}\right)^{d_1}\left(\hat{Y}^W_{V_i,\mathrm{tr}}\right)^{d_2}\left(\hat{Z}^W_{V_i,\mathrm{tr}}\right)^{d_3} \end{cases} \quad d_i \in \mathbb{N}_0, \sum_i d_i \leq d_{\mathrm{all}}. \tag{6.4}$$

where $\hat{P}^W_{V_i,\mathrm{tr}} = \left[\hat{X}^W_{V_i,\mathrm{tr}}, \hat{Y}^W_{V_i,\mathrm{tr}}, \hat{Z}^W_{V_i,\mathrm{tr}}\right]^T$ is a train viewpoint's position in the world space, $b_{ij,u}\left(\cdot\right)$ and $b_{ij,v}\left(\cdot\right)$ are coefficients, and d_{all} is the maximal degree of the regression function. If we set $d_{\mathrm{all}} = 3$, then for each 2D–3D correspondence in a single view, we have $20 \times 2 = 40$ coefficients. Suppose we have 80 control points and 25 train viewpoints, then we have totally $40 \times 80 \times 25 = 8 \times 10^4$ coefficients, which requires 79 KB if we store them as float type. If we set $d_{\mathrm{all}} = 4$, then we need 137 KB storage space.

Wientapper et al. [10] employed a similar yet more comprehensive regression model, considering additionally another two variables u_{ij} and v_{ij} that represent normalized observed virtual point positions. Therefore, for each 2D–3D correspondence in a single view they have $56 \times 2 = 112$ coefficients when $d_{\mathrm{all}} = 3$, and $126 \times 2 = 252$ coefficients when $d_{\mathrm{all}} = 4$. With 80 control points and 25 train viewpoints, they need 219 KB and 493 KB storage spaces, respectively. Therefore, in terms of storage for distortion coefficients, the above state of the art are more efficient than using warping maps.

However, solving those polynomial regression models requires a massive number of captured pictures from the calibration camera. Wientapper et al. [10] captured two sequences containing 1193 and 2251 frames, respectively, whereas Deng et al. [18] stated that 500 sample photos are enough. Note that the photos are taken at different individual train viewpoints. In contrast, we have sampled far fewer train viewpoints, which is enough for our distortion correction based on interpolation and less demanding on the camera moving device. For example, some cheaper camera tripods can also be employed instead of motored linear stages.

We briefly summarize the above three distortion models' strength and weakness in Table 6.7. Indeed, there is not a perfect approach to eliminate the optical distortion existing in automotive AR-HUDs. Practical solutions should be selected according to concrete scenarios.

Table 6.7: Comparison among various distortion models for AR-HUD calibration.

Model	**Camera distortion model**	**Warping maps**	**Polynomial regression**
Type	Parametric	Nonparametric	Parametric
Data size dependency	Number of viewpoints, order of formula	Number of viewpoints, virtual image resolution	Order of formula
Reflection of virtual image detail	Low	High	Low
Requirement on storage & computation	Low	High	Low
Functionality on interpolation among viewpoints	Low	High	High
Required viewpoint sample size	Medium	Small	Large

7. Conclusion & Outlook

This research aimed to identify different factors that influence AR-HUDs' calibration and develop our own methodologies for this purpose. The principles are based on the pinhole camera model because of the perspective projection mechanism in the image formation and receiving. Both the simulation and experimental results are provided in qualitative and quantitative aspects.

We simulated error sources, such as 2D/3D detection errors, tolerance of viewpoint tracking and optical distortion. We have found out there exist threshold values for some of these quantities, under which the calibration accuracy can be guaranteed. We have also taken the number of control points and the HUD's opening angles into account. The results indicate that the calibration accuracy also relies on target characteristics (if it is used) and the HUD's specific parameters.

We proposed and proved new novel calibration schemes for automotive AR-HUDs, expecting to enhance the time-efficiency and reduce the target complexity while maintaining a high level of precision. First, we presented our concepts with an evolution that includes the manual and automatic implementations. The transformations between the pre-defined world coordinate system and other reference frames, i.e. viewpoint space, HUD-FOV space and virtual image, are accurately restored. Our automatic implementations are based on pattern recognition. They can exploit more 2D–3D correspondences than the manual one while consuming far less time. We have successfully developed various automatic calibration solutions, including the schemes using a sizable chessboard target, a piece of printed paper target and even no target. Particularly in the target-free calibration, we applied the SIFT algorithm for feature extraction and epipolar constraint to combine multiple views. Finally, the calibration period for an AR-HUD is reduced to less than 5 min. Meanwhile, the implementation process has become more and more friendly to the operator, mainly thanks to the lowered target complexity and freed human labor.

The calibration of AR-HUDs is highly dependent on the viewpoint positions. Some involved coordinate systems, e.g. the 3D HUD-FOV space H_i, varies together with the viewpoint. We have also pointed out that the virtual image space I_i is view-dependent, which is a challenge to the conventional assumption of static virtual images. To deal with multiple views, we have presented the selection of an eye box and corresponding interpolation methods so that the calibration can cover any reasonable viewpoint. The interpolation concept has been validated in the evaluation phases under all the proposed schemes, both qualitatively and quantitatively.

To cope with the nonlinear optical distortion caused by the HUD optics and curved windshield, we have developed a useful tool, i.e. warping maps. It is reconstructed from reprojection errors in the validation phases and appears in a nonparamatric form. Indeed, it compensates not only the distortion, but also potential offsets in the estimated intrinsic and extrinsic matrices. Notice that though this tool has shown a convincing performance, some residual distortion still exists. A deeper understanding of AR-HUD's optical distortion and the development of uncomplicated yet robust rectification models can be a future research topic.

Our experimental data from the lab are carefully analyzed. Compared to state of the art, our calibration accuracy is competitive, even though the target-free scheme sacrifices a bit on this aspect. In summary, the calibration concepts we have proposed are diverse and highly applicable in the automobile industry.

There are still some open points in this work waiting for solutions. In our calibration concepts, the viewpoint is referred to as the middle point between the driver's eyes. However, there are, after all, two eyes on a human face. Therefore, rigorously speaking, it is worth considering the human binocular vision in the AR-HUD calibration as a future topic.

It is also notable that none of the proposed calibration methods is perfect. For example, when we use a conventional calibration target standing in front of the vehicle, it must be large enough to cover the HUD-FOV. When we use a smaller paper target laid on the windshield, it can easily cover the HUD-FOV. However, when we calibrate the transformation between the 3D sensor and smartphone camera, this small target can only occupy an extremely limited area in the joint FOV. In this case, the calculated transformation matrix may be biased. When we calibrate an AR-HUD in a target-free manner, selecting an appropriate frontal scene is an essential prerequisite; otherwise, too few feature points cannot fulfill the demand. Practically speaking, there is not "the best" choice for the automotive industry, i.e. the selection of the calibration scheme should be based on the specific scenario. Ideally, we recommend that factories or workshops adopt one of the target-based approaches, whereas the customer should be able to choose the target-free approach as a more cost-effective option.

Acknowledgement

My work in this Ph.D. thesis was inspired by many others who have offered me kind help. Therefore the word “we” was used instead of the term “I.”

I want to thank Mercedes-Benz AG, Sindelfingen, since they provided me their cars and equipment for AR-HUD calibration experiments. Then, I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my Ph.D. study!

Prof. Wilhelm Stork at Karlsruhe Institute of Technology (KIT) steered me throughout this research. He provided me with guidance and suggestions on how to form my work into a valuable and clear scientific profile. Prof. Uli Lemmer at KIT reviewed this thesis very carefully. Dr. Marc Necker at Mercedes-Benz AG, Sindelfingen managed my research at the company. As the team leader, he gave me the chance to do the job in a very comfortable atmosphere and helped organize my working plan with efforts. Dr. Janis Werner, Philipp Pitzer and Adam Schatton, colleagues at Mercedes-Benz, led me to solve many technical problems with the cars and other equipment. Steffen Tacke, my department leader at Mercedes-Benz, gave me critical feedback on my project presentations. Martin Keppler, a colleague in the department, collaborated with me on several relevant patent writing.

Felicia Ruppel, Kai Wu, Lu Guo, Lukas Jaeckle and Christian Eissler shared much coding knowledge in *Python* and C++. Jonas Haeling helped me much in brainstorming and conference paper revision. Gordana Bojovic always gave me much instant support in the company, especially in many administrative affairs.

I also would like to thank Rui Ma at KIT for sharing his experience in thesis writing. Thank Ye Liu for her patience, encouragement, and the precious friendship we have throughout the years.

Finally, I shall give many thanks to my parents in China, who showed me, invaluable love, even at a far distance.

Bibliography

[1] H. Okumura, "Human centric AR & VR display and interface technologies for automobile," *IEEE Consumer Electronics Magazine*, vol. 8, no. 5, pp. 60–61, 2019.

[2] J. H. Iavecchia, H. P. Iavecchia, and S. N. Roscoe Illiana, "Eye accommodation to head-up virtual images," *Human Factors*, vol. 30, no. 6, pp. 689–702, 1988.

[3] D. C. Foyle, A. J. Ahumada, J. Larimer, and B. T. Sweet, "Enhanced/synthetic vision systems: Human factors research and implications for future systems," *SAE Transactions*, pp. 1734–1741, 1992.

[4] D. P. Burch and M. Braasch, "Enhanced head-up display for general aviation aircraft," in *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, IEEE, vol. 2, 2002, pp. 11C6–11C6.

[5] L. J. Prinzel III and M. Risser, "Head-up displays and attention capture," 2004.

[6] T. Nojima and H. Kajimoto, "A study on a flight display using retro-reflective projection technology and a propeller," in *CHI'08 Extended Abstracts on Human Factors in Computing Systems*, 2008, pp. 2721–2726.

[7] S. Patterson, J. Farrer, and R. Sargent, "Automotive head-up display," in *Automotive Displays and Industrial Illumination*, International Society for Optics and Photonics, vol. 958, 1988, pp. 114–123.

[8] J. A. Betancur, J. Villa-Espinal, G. Osorio-Gómez, S. Cuéllar, and D. Suárez, "Research topics and implementation trends on automotive head-up display systems," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 12, no. 1, pp. 199–214, 2018.

[9] Y. Hwang, B.-J. Park, and K.-H. Kim, "Effects of augmented-reality head-up display system use on risk perception and psychological changes of drivers," *ETRI Journal*, vol. 38, no. 4, pp. 757–766, 2016.

[10] F. Wientapper, H. Wuest, P. Rojtberg, and D. Fellner, "A camera-based calibration for automotive augmented reality head-up-displays," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013, pp. 189–197.

[11] D. Wagner, M. Schneider, F. Dross, S. Langer, S. Zeidler, T. Ganz, and U. Lemmer, "Impact study of windshield geometry on the subjective customer perception for augmented reality head-up displays (AR HUD)," in *SID Symposium Digest of Technical Papers*, Wiley Online Library, vol. 51, 2020, pp. 254–257.

[12] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[13] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[14] M. Biswas and S. Xu, "World fixed augmented-reality HUD for smart notifications," in *SID Symposium Digest of Technical Papers*, Wiley Online Library, vol. 46, 2015, pp. 708–711.

[15] A. Hosseini, D. Bacara, and M. Lienkamp, "A system design for automotive augmented reality using stereo night vision," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, IEEE, 2014, pp. 127–133.

[16] K. Ueno and T. Komuro, "Overlaying navigation signs on a road surface using a head-up display," in *2015 IEEE International Symposium on Mixed and Augmented Reality*, 2015, pp. 168–169.

[17] C. Yoon and K.-H. Kim, "Augmented reality information registration for head-up display," in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2015, pp. 1135–1137.

[18] N. Deng, Y. Zhou, J. Ye, and X. Yang, "A calibration method for on-vehicle AR-HUD system using mixed reality glasses," in *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2018, pp. 541–542.

[19] M. Tuceryan, Y. Genc, and N. Navab, "Single-point active alignment method (SPAAM) for optical see-through hmd calibration for augmented reality," *Presence: Teleoperators & Virtual Environments*, vol. 11, no. 3, pp. 259–276, 2002.

[20] Y. Itoh and G. Klinker, "Interaction-free calibration for optical see-through head-mounted displays based on 3D eye localization," in *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, Mar. 2014, pp. 75–82.

[21] ——, "Light-field correction for spatial calibration of optical see-through head-mounted displays," *IEEE Transactions on Visualization & Computer Graphics*, vol. 21, no. 4, pp. 471–480, Apr. 2015.

[22] ——, "Simultaneous direct and augmented view distortion calibration of optical see-through head-mounted displays," in *2015 IEEE International Symposium on Mixed and Augmented Reality*, Sep. 2015, pp. 43–48.

[23] R. Haeuslschmid, Y. Shou, J. O'Donovan, G. Burnett, and A. Butz, "First steps towards a view management concept for large-sized head-up displays with continuous depth," in *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2016, pp. 1–8.

[24] X. Gao, J. Werner, M. Necker, and W. Stork, “A calibration method for automotive augmented reality head-up displays using a chessboard and warping maps,” in *Twelfth International Conference on Machine Vision (ICMV 2019)*, W. Osten and D. P. Nikolaev, Eds., International Society for Optics and Photonics, vol. 11433, SPIE, 2020, pp. 787–794.

[25] M. Klemm, F. Seebacher, and H. Hoppe, “Non-parametric camera-based calibration of optical see-through glasses for AR applications,” in *Cyberworlds (CW), 2016 International Conference on*, IEEE, 2016, pp. 33–40.

[26] ——, “High accuracy pixel-wise spatial calibration of optical see-through glasses,” *Computers & Graphics*, vol. 64, pp. 51–61, 2017.

[27] X. Gao, J. Werner, M. Necker, and W. Stork, “A calibration method for automotive augmented reality head-up displays based on a consumer-grade mono-camera,” in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 4355–4359.

[28] J. Wang, F. Shi, J. Zhang, and Y. Liu, “A new calibration model of camera lens distortion,” *Pattern Recognition*, vol. 41, no. 2, pp. 607–615, 2008.

[29] Q. Zhu, “A technique of camera calibration using single planar calibration image,” in *Image and Signal Processing (CISP), 2012 5th International Congress on*, IEEE, 2012, pp. 824–827.

[30] J. Bloomenthal and J. Rokne, “Homogeneous coordinates,” *The Visual Computer*, vol. 11, no. 1, pp. 15–26, 1994.

[31] J. J. Craig, *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.

[32] E. Hecht and K. Lippert, *Optik*, ser. De Gruyter Studium. De Gruyter, 2018, ISBN: 9783110526707.

[33] N. A. Dodgson, “Variation and extrema of human interpupillary distance,” in *Stereoscopic Displays and Virtual Reality Systems XI*, International Society for Optics and Photonics, vol. 5291, 2004, pp. 36–47.

[34] M. J. Magee and J. K. Aggarwal, “Determining vanishing points from perspective images,” *Computer Vision, Graphics, and Image Processing*, vol. 26, no. 2, pp. 256–267, 1984.

[35] R. G. Willson and S. A. Shafer, “Perspective projection camera model for zoom lenses,” in *Optical 3D Measurement Techniques II: Applications in Inspection, Quality Control, and Robotics*, International Society for Optics and Photonics, vol. 2252, 1994, pp. 149–158.

[36] K. Naus and A. Makar, “Dynamic perspective projection for presentation of the geometrical information about the geographical environment,” *space*, vol. 3, no. 2, p. 1, 2002.

[37] Y. I. Abdel-Aziz, “Direct linear transformation from comparator coordinates into object space in close-range photogrammetry,” in *Proceedings of the ASP Symposium on Close-Range Photogrammetry, 1971*, American Society of Photogrammetry, 1971, pp. 1–18.

[38] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi, “Revisiting the PnP problem: A fast, general and optimal solution,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2344–2351.

[39] C. Chatterjee and V. P. Roychowdhury, “Algorithms for coplanar camera calibration,” *Machine Vision and Applications*, vol. 12, no. 2, pp. 84–97, 2000.

[40] K. Sirisantisamrid, K. Tirasesth, and T. Matsuura, “A technique of camera calibration using single view,” in *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, IEEE, 2011, pp. 1486–1490.

[41] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[42] F. Moreno-Noguer, V. Lepetit, and P. Fua, “Accurate non-iterative $O(n)$ solution to the PnP problem,” in *2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–8.

[43] E. Marchand, H. Uchiyama, and F. Spindler, “Pose estimation for augmented reality: A hands-on survey,” *IEEE transactions on visualization and computer graphics*, vol. 22, no. 12, pp. 2633–2651, 2015.

[44] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[45] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[46] D. V. Papadimitriou and T. J. Dennis, “Epipolar line estimation and rectification for stereo image pairs,” *IEEE transactions on image processing*, vol. 5, no. 4, pp. 672–676, 1996.

[47] T. S. Huang and O. D. Faugeras, “Some properties of the E matrix in two-view motion estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 12, pp. 1310–1312, 1989.

[48] N. Jacobson, *Basic algebra I.* Courier Corporation, 2012.

[49] R. I. Hartley, “Cheirality invariants,” in *Proc. DARPA Image Understanding Workshop*, vol. 3, 1993.

[50] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

[51] ——, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[52] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge university press, 2007.

[53] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[54] L. S. Díaz, "Optical aberrations in head-up displays," M.S. thesis, Technische Universität München, 2005.

[55] K. Blankenbach, "Requirements and system aspects of AR-head-up displays," *IEEE Consumer Electronics Magazine*, vol. 8, no. 5, pp. 62–67, 2019.

[56] X. Gao, M. Necker, and W. Stork, "A low-complexity yet accurate calibration method for automotive augmented reality head-up displays," in *Thirteenth International Conference on Machine Vision*, International Society for Optics and Photonics, vol. 11605, 2021, 116050B.

[57] X. Gao, K. Wu, M. Necker, W. Stork, A. Jadid, and G. Klinker, "A target-free calibration method for automotive augmented reality head-up displays," in *Thirteenth International Conference on Machine Vision*, International Society for Optics and Photonics, vol. 11605, 2021, p. 116051V.

[58] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 10, pp. 965–980, 1992.

[59] P. Drap and J. Lefèvre, "An exact formula for calculating inverse radial lens distortions," *Sensors*, vol. 16, no. 6, p. 807, 2016.

[60] R. I. Hartley, "Self-calibration from multiple views with a rotating camera," in *European Conference on Computer Vision*, Springer, 1994, pp. 471–478.

[61] L. Krüger and C. Wöhler, "Accurate chequerboard corner localisation for camera calibration," *Pattern Recognition Letters*, vol. 32, no. 10, pp. 1428–1435, 2011.

[62] C.-C. Lee and C.-C. Kuo, "Optical coatings for displays and lighting," in *Optical Thin Films and Coatings*, Elsevier, 2018, pp. 565–594.

[63] L. J. Hornbeck, "Digital light processing for high-brightness high-resolution applications," in *Projection Displays III*, International Society for Optics and Photonics, vol. 3013, 1997, pp. 27–40.

[64] G. Sauer, *Laminated glass windscreen intended to be used at the same time as a hud system reflector*, US Patent App. 11/057,161, Jun. 2005.

[65] M. Marcus, "Simultaneous head-up display windshield wedge-angle and layer-thickness measurements," *SPIE Newsroom*, 2016.

[66] G. Peng and M. J. Steffensmeier, *Head up display having a combiner with wedge lenses*, US Patent 7,570,430, Aug. 2009.

[67] P. Thomas and W. David, "Augmented reality: An application of heads-up display technology to manual manufacturing processes," in *Hawaii International Conference on System Sciences*, 1992, pp. 659–669.

[68] I. T. Feldstein, A. Güntner, and K. Bengler, “Infrared-based in-vehicle head tracking: A prototype for tracking drivers’ head movements in real time,” *Procedia Manufacturing*, vol. 3, pp. 829–836, 2015.

[69] M. Kutila, M. Jokela, G. Markkula, and M. R. Rué, “Driver distraction detection with a camera vision system,” in *2007 IEEE International Conference on Image Processing*, IEEE, vol. 6, 2007, pp. VI–201.

[70] B. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[71] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, “Complete solution classification for the perspective-three-point problem,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[72] G. Jiang and L. Quan, “Detection of concentric circles for camera calibration,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, IEEE, vol. 1, 2005, pp. 333–340.

[73] C. Portalés, P. Casanova-Salas, S. Casas, J. Gimeno, and M. Fernández, “An interactive cameraless projector calibration method,” *Virtual Reality*, vol. 24, no. 1, pp. 109–121, 2020.

[74] X. X. Lu, “A review of solutions for perspective-n-point problem in camera pose estimation,” in *Journal of Physics: Conference Series*, vol. 1087, 2018, p. 052 009.

[75] Z. Zhao, D. Ye, X. Zhang, G. Chen, and B. Zhang, “Improved direct linear transformation for parameter decoupling in camera calibration,” *Algorithms*, vol. 9, no. 2, p. 31, 2016.

Appendix

Abbreviations throughout the thesis

Abbrev.	**Meaning**	**Abbrev.**	**Meaning**
2D	two-dimensional	3D	three-dimensional
ADAS	advanced driver assistance systems	AR	augmented reality
AR-HMD	augmented reality head-mounted display	AR-HUD	augmented reality head-up display
AV	autonomous vehicle	CAD	computer-aided design
DLP	digital light processing	DLT	direct linear transformation
DMD	digital micromirror device	DoF	degree of freedom
DoG	difference of Gaussians	FOV	field of view
Full HD	full high definition	GNSS	global navigation satellite system
HMD	head-mounted display	HMI	human-machine interface
HUD	head-up display	HUD-FOV	field of view of head-up display
ICP	iterative closest point	IP	Internet protocol
KB	kilobyte	LCD	liquid-crystal display
LiDAR	light detection and ranging	LUT	look-up table
MB	megabyte	MEMS	microelectromechanical system
PEP	polyethylene propylene	PGU	picture generation unit
PnP	perspective-n-point	RGB-D	red, green, blue and depth
RMSE	root-mean-square error	ROI	region of interest
SfM	structure-from-motion	SIFT	scale-invariant feature transform
SPAAM	single point active alignment method	STD	standard deviation
SVD	singular value decomposition		