# A TOOL FOR SYNTHETIC EVALUATION OF ACTIVE CALIBRATION ALGORITHMS

A Thesis Submitted to
the Graduate School of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

**MASTER OF SCIENCE**

**in Computer Engineering**

by
Buğrahan DÖNMEZ

October 2022
İZMİR

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Assoc. Prof. Dr. Mustafa ÖZUYSAL, for his guidance, encouragement and teaching me the concepts of computer vision. Furthermore, sharing his industrial and academic experiences has given me a priceless perspective to the outside world.

# ABSTRACT

## A TOOL FOR SYNTHETIC EVALUATION OF ACTIVE CALIBRATION ALGORITHMS

To calibrate a camera, the choice of poses is very important and different angled poses can increase accuracy. Gathering those poses needs expert intuition in order to constrain all parameters accurately. There are various tools to help users calibrate the camera with its guidance.

In this study, two successful calibration tools are tested. Both of them guide the user interactively to obtain the best poses. The first method tries to avoid singular poses and captures the poses that reduce the uncertainty of calibration. The second method uses a different approach. It uses the current calibration state to suggest the next pose. In the end, it verifies the parameters with the specified ones by the user.

To test these two methods, ground truth data is needed. The ground truth data is obtained with the help of a 3D modeling program. The suggested poses are generated also with the modeling program and knowing the ground truth camera parameters given in the program, the results of the tools are compared.

# ÖZET

## AKTİF KALİBRASYON ALGORİTMALARININ SENTETİK DEĞERLENDİRMESİ İÇİN BİR ARAÇ

Kamerayı kalibre etmek için poz seçimleri çok önemlidir ve farklı açılardan pozlar toplayabilmek, kalibrasyonun kesinliğini arttırabilir. Kamera parametrelerini doğru şekilde kalibre etmek için farklı pozlara ihtiyaç vardır ve bu iş için alanında uzman kişiler gerekebilir. Birçok araç, yönlendirme ile kamera kalibrasyonu için kullanıcılara destek olmaktadır.

Bu çalışmada, iki başarılı kalibrasyon aracı test edilmiştir. İkisi araç da, en iyi pozları toplayabilmek için interaktif şekilde kullanıcıları yönlendirmektedir. İlk yöntem, tekil pozlardan kaçınmaya çalışır ve kalibrasyon belirsizliğini azaltan pozları yakalar. İkinci yöntem farklı bir yaklaşım kullanır. Bir sonraki pozu önermek için mevcut kalibrasyon durumunu kullanır. Sonunda kullanıcı tarafından belirtilen parametrelerle parametreleri doğrular.

Bu iki yöntemi test etmek için temel gerçek verilerine ihtiyaç vardır. Temel gerçek verileri bir 3B modelleme programı yardımıyla elde edilir. Modelleme programı ile de önerilen pozlar oluşturulmuştur ve programda verilen temel gerçek kamera parametreleri bilinerek araçların sonuçları karşılaştırılmıştır.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation and Problem Definition

In this thesis, the aim is to develop an evaluation process that tests guided calibration tools with ground truth camera parameters created with a modeling tool. The developed evaluation system is used as a ground truth creator and provides repeatable test cases.

Camera calibration is a field that is important for every vision problem. Camera calibration means obtaining the required camera parameters. Every camera has its own characteristics and the camera calibration process is the way of estimating these characteristics. These parameters are required in order to calculate the relation between the 3D world and the 2D world.

The created process can be used in all camera calibration systems. All the cameras should be calibrated well in order to work accurately. Dealing with 3D world measurements requires both the intrinsic and extrinsic parameters found correctly. These parameters are the ones to describe the mathematical relationship between the 3D coordinates of a point and its 2D projection onto the image plane.

The intrinsic parameters can also be called internal parameters. These parameters define the camera. For instance focal length and lens distortion. To make a good prediction, the accuracy of the focal length must be very definite.

The other parameter group is called extrinsic parameters and they are also known as external parameters. Extrinsic parameters define the 3D world relation between the camera and the outer world.

The accuracy of the calibration is very important. For instance, in autonomous car systems, the camera placed in the car should be accurate. Accuracy means how close the results are to the ground truth. Precision means how close the results are to each other. Camera accuracy means the camera should have low or no distortion. High precision describes the same output quality for any kind of environment. High resolution describes the camera should be able to decipher the smallest possible change. Hence, the camera should be well calibrated in order to have high accuracy and high precision.

The parameter calculation process was always a concern and much research are done in this field. However, there are still some issues with finding the correct parameters. There are different cases of camera calibration problems. When the number of points increases, the number of solutions increases [11]. Another problem is choosing the right

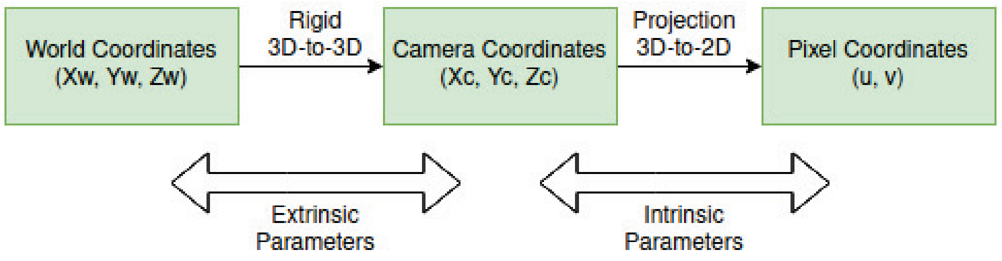


Figure 1.1: Usage of parameters and conversion between planes

orientations for the checkerboard. Even the most educated person can make mistakes. To get rid of human errors, calibration tools started to increase. These tools require only a checkerboard with the desired shapes and a camera. The suggestion mechanism generates new poses and asks users to move the checkerboard to that generated pose [24][23]. With each step, both the intrinsic and the external parameters update themselves. Another problem is choosing the right pattern. Board patterns affect the accurate estimation of intrinsic and extrinsic parameters. Point and corner detection accuracy increase the accuracy of the calibration [13].

The problems above can be solved but still needs to be checked with the ground truth data in order to be sure about the accuracy. With the help of the proposed modeling process, the accuracy can be tested and calibrated cameras can be more reliable. Furthermore, accurately calibrated cameras can be used in a variety of scenes for different scenarios. For instance, in autonomous driving systems, cameras must be accurate. Autonomous driving cameras mostly have too much distortion. Lens distortion means deformations that occur in the image produced by the camera lens. Distortion can be seen in the images as the lines appear curvy or bent. Autonomous driving cameras need to estimate distance calculations and geometric features accurately [18].

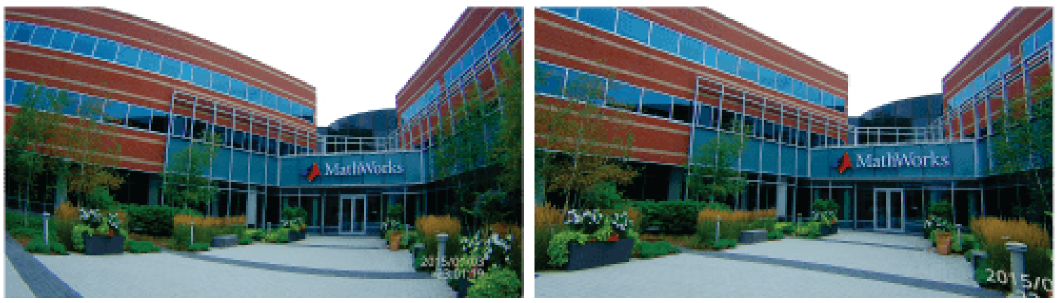


Figure 1.2: Distorted and Undistorted Image (Source:[19])

Another scenario is the advertisement insertion system. Synthetic advertisements can be seen in TV programs and internet videos. A camera calibration system is required in these systems and different calibration techniques are used in order to estimate camera parameters. Also, estimated parameters should be accurate to make ads more realistic [15]. The last example is about football and multiple cameras. The system gives the positions of the player in real-time. First, it uses a single camera as a source and then uses multiple cameras to get more data. Using multiple cameras requires each camera to be calibrated accurately. This accuracy is used in merging data from different angles and increases the correctness of the position of the player [29].

## 1.2 Literature on Calibration Tools

Camera calibration can be considered in 3D computer vision. This is the process of determining the physical characteristics of the camera (intrinsic parameters) and the location and direction of the camera in a 3D world coordinate system (extrinsic parameters).

Almost every computer vision algorithms strictly depend on the accuracy of the calibration. Moreover, the calibration process must be done after each setup change. Even if the location of the camera stays the same, the truthfulness of the intrinsic parameters may vary. The most known approach to calibrating the camera is Zhang's method [30]. This method is done by calculating the relationship between the world and image plane corners with a planar calibration pattern.

Nonetheless, there can be degenerate poses and this leads to inaccurate calibration. Due to this reason, inexperienced people cannot gather good quality poses, and even the researchers working in this area encounter this problem [24]. There are some important constraints regarding the position of the checkerboard. The position of the board in front of the camera, rotation, size, tilt angle, and shift is one of the most necessary constraints that are really important while gathering efficient poses.

There has been researching on to determine how to use camera orientation constraints. Research about angular spread showed us that at least spread is necessary to decrease the error in focal length [26]. Also, Sturm and Maybank [25] found a relation between the principal point and focal length. The focal length cannot be calculated if the patterns in the frames are parallel to the image plane.

With respect to the research above, a calibration tool was proposed by Rojtberg and Kuijper [24]. In this method, they create optimal checkerboard poses while trying to not get degenerate poses. Using the research mentioned above, a pose generation process is created.

In contrast to the work created by Rojtberg and Kuijper [24], there are other methods that work iteratively rather than having predefined pose generation steps. The method presented by Richardson, Strom and Olson [23] uses two process to generate poses and calibrate the camera. The first is bootstrapping an initial camera parameters by calculating the focal length stabilities and the second one is reducing the uncertainty over pixels iteratively.

Another method that works iteratively is proposed by Peng and Sturm [21]. This method tries to reduce the intrinsic uncertainty step by step by generating the next pose. The next pose is determined by the least uncertain parameters using previously captured poses. Another contribution of the proposed method is to quantify intrinsic parameters and calibration by using corner uncertainty.

There are also some different approaches in calibration tools alternative to iterative ones. The work proposed by Ren and Hu [22] uses initial pose sets that are generated among candidate pose sets. Their proposed score function is used when determining which pose set should be selected. After selecting a pose set for the calibration process, poses in the set are displayed step by step to the user with instructions in order to guide the user to move the checkerboard pattern. Each step reduces the reproduction error and intrinsic parameter estimation variance.

In order to calibrate the camera, tools and algorithms need data. Creating synthetic data is also an option. Chen and Little [5] calibrates sports camera with synthetic data. They generate poses with three different parameters and feed them to a GAN network and refine poses.

Recent works showed us that generating synthetic data prevent us from getting unwanted noise and camera effects. Using synthetic data increase the size and variability. To get rid of sensor effects, a work is proposed. These sensor effects are chromatic aberration, blur, exposure, noise, and color temperature [4].

Using synthetic data is useful but it should be tested with image processing algorithms. To see if they are working accurately, a paper is proposed. The tested field is the lane tracking problem. It is a daily problem for the autonomous vehicle industry. Ground truth data is needed to test everything repeatedly. It is possible to compare the results of synthetic data with the ground truth data after tests [27].

Synthetic data can mimic real-world data. Semantic segmentation tasks depend on real-world images to segment things correctly. With the created synthetic data, data can be increased without using real images. In a work proposed lately, real-world images were replaced by synthetic data for foggy images. Segmentation models are trained with synthetic data [8].

## 1.3 Methods Used

The evaluation process system is based on a 3D modeling tool. The modeling tool used in this system is Blender [6]. Blender is a free and open-source 3D software. It is used for creating 3D models, film, animations, visual effects, and video games. Being open-source and free was one of the reasons to choose Blender. The ease of use is also the other reason.

To test the calibration tools in real-time, desired checkerboard patterns are printed on cardboard. Testing was done using these cardboards. There are various cardboard designs. Three of them are used in this work. These are AprilTags [28], ArucoMarkers [**?** ], and basic checkerboard patterns.

The first calibration tool is implemented using Python. The software uses PyQt as a user interface. After gathering the necessary poses, it uses OpenCV [3] library to calibrate the camera. The calibration function of OpenCV uses Zhang's method to calibrate the camera.

The second calibration tool is implemented using MatLab [19]. The user interface of AprilCal is designed to show generated poses to the user. Users are allowed to move cardboard regarding target poses. The interface also prompts some text about instructions. Users can read them and continue the process.

In order to create the generated poses with Blender, the steps of pose suggestion were applied in the modeling software. For instance, if the calibration tool suggests cardboard with an orientation that is tilted on the principal axis by , the same orientation is created in the Blender scene. Then, exporting this model gives a pose with ground truth.

Using the Blender generated data that is created in the way given above, the next step is testing these poses with error metrics. The used error metrics are implemented using Python. These error metrics are Maximum Reproduction Error, triangulation, camera parameter changes over iteration, and keypoint matching. All of these error metrics are implemented with the help of OpenCV and NumPy [9]. Result are plotted using MatPlotLib [12].

## 1.4 Organization of the Thesis

The organization of the thesis is as follows:

- Chapter 2 gives the mathematical background that is necessary to understand this field and work. This chapter consists of some of the basic vision concepts, calibration algorithms, and linear algebra operations.

- In Chapter 3, calibration tools that are tested in this thesis are explained. Their results are given.

- In Chapter 4, used modeling tool and implementations are explained. Evaluation metrics are given in this chapter. A comparison of the calibration tools with the ground truth data is given in this chapter.

- Chapter 5 gives the summary of the obtained results in this thesis. At the end of the chapter, a discussion and future works section are given.

# CHAPTER 2

# THEORETICAL BACKGROUND

## 2.1 Introduction

In this chapter, the mathematical background that is necessary to understand the work represented in this thesis is presented. To follow the mathematical derivations given in the work, one needs to understand these basic mathematical concepts. Section 2.2 presents the main aspects and notation of projective geometry. These ideas are the core elements of a multiple view geometry. Projective transformations of the 2D world are introduced. Affine and similarity transformations are introduced in this section. How to get affine and metric properties from an image is focused on. After introducing 2D world concepts, the next concepts are projective geometry of 3D world space. The new concept compared to 2D geometry here is the infinity plane and the absolute conic.

Section 2.3 concentrates on the geometry of the camera. This section describes the 3D to 2D projection. The basic camera matrix P is introduced. It is a 3x4 matrix. Matrix P is responsible for the mapping of homogeneous coordinates of 3D world space to homogeneous coordinates of 2D image space. The camera parameters are also introduced in this section. Both the intrinsic and extrinsic parameter derivations are presented in this section. The internal camera parameters such as focal length and principal point are obtained from matrix P. Estimation of the matrix P is described in this section. The derivation comes from the corresponding world and image coordinates that are introduced in the previous section.

In Section 2.4, the geometry of two perspective views is covered. The relation between the two perspective views is explained. There are two cameras and the transition from one view to another view is done using matrices $P$ and $P'$. The coordinates in these two perspective view correspond. The reason behind this is these points are in the same 3D world space. To obtain these coordinates and relationship between them, there are three main topics, correspondence geometry, camera geometry and scene geometry. To explain the first topic, the epipolar geometry is described in this section. Epipolar line is a line that passes from one point to another point in different views. These points are correspondent. The epipolar line depends on the camera. To understand epipolar geometry, the fundamental matrix concept should be explained and in this section, the 3x3 fundamental matrix F is described. The computation of the fundamental matrix F comes from $P$ and $P'$. The vice versa is also explained in this section.

Section 2.5 describes the triangulation concept. Triangulation is the method of finding a point in 3D space based on its projections onto two or more images in computer vision. In order to solve this problem, camera projections of 3D and 2D should be known. These projections is mentioned in the previous sections. To use triangulation, accurate camera matrices are necessary. Sometimes this concept is also mentioned as reconstruction.

These concepts are very detailed and to understand deeper, the reader should refer to the [10]. More detailed derivations and definitions are explained in this reference.

## 2.2 Projective Geometry

This section presents the 2D and 3D projective geometry properties. There are some particular representations used in this field. A point in the image plane can be represented by the coordinates like $(x, y)$ in 2D image plane. Also, the 2D image plane can be considered as vector space. In vector space, there are vectors and a point 2D image plane can be called a vector. Hence, a point is identified as a vector.

There is a specific notation that is used in the representation of a vector. Vectors are represented by $x$ and this means that this is a column vector. The transpose of this vector gives us the row vector, $x^T$. To represent the point in the vector space, column vector notation is used rather than row vector notation. In the derivations, $x = (x, y)^T$ is used to represent vectors.

In projective geometry, lines are represented by equations. To represent a line, an equation such as $ax + by + c = 0$ is used. This line can also be represented with a column vector notation such as $(a, b, c)^T$.

In the representation of the lines and vectors, homogeneous representation is used. To describe a homogeneous point in 3D vector space, $x = (x_1, x_2, x_3)^T$ is used. To represent this point in the 2D vector space, $x = (x_1/x_3, x_2/x_3, )^T$ is used. One advantage of using homogeneous coordinate system is that the representation of infinite points can be done using it. Basically, $z = 0$ means that the point is at infinity.

In order to determine whether a point lies on the line, the equation below should be satisfied.

$$x^T l = 0$$

The equation above shows us that the point $x$ lies on the line $l$ if and only if the scalar product of the point $x$ and the line $l$ equals 0.

The concept of degrees of freedom is also important while discussing projective geometry. In order to represent a point in the image plane, two main properties must be satisfied. These two properties are $x$ and $y$ coordinates. Also, a line should have two properties same as a point. Hence, it is dependent on these two values and it has two degrees of freedom.

Given two lines $l = (a, b, c)^T$ and $l^{'} = (x, y, z)^T$. To find the intersection of these lines, one must satisfy the equation following.

$$x = l \times l^{'} \tag{2.1}$$

Equation 2.1 tells that the intersection of two lines is the point $x$. The line passes over two points $x$ and $x^T$ is represented with Equation 2.2.

$$l = x \times x^{'} \tag{2.2}$$

This means that the lines go through the points $x$ and $x^T$.

To represent finite points in the image plane, $x_3 \neq 0$ for the homogeneous vector $x = (x_1, x_2, x_3)^T$. If $x_3 = 0$, the vector is called ideal points or points at infinity. Regarding these, the line $l$ can intersect the line $l_\infty$ in the ideal point.

As seen in the previous equations, points and lines are interchangeable. The roles of points and lines can be interchanged. For instance, $l^T x = 0$ can be interchanged with $x^T l = 0$. The positions of line and point are swapped. This concept is called *duality principle* and the related point and line are *dual* to each other.

Mapping from point to point means projectivity. It is invertible. Projectivity is also known as homography or projective transformation. Let's call a mapping $h$ is a projectivity. This is true if and only if there is an invertible $3x3$ matrix H. The transformation can be represented as follows.

$$\begin{bmatrix} x_1^{'} \\ x_2^{'} \\ x_3^{'} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2.3}$$

To represent the Equation 2.3 more simpler, $x^{'} = Hx$. Regarding the point transformations, a line transformation can be represented as $l^{'} = H^{-T} l$.

There are some main projective transformations. Transformations work by changing the values of the vector but sometimes the properties of the vector stay unchanged. This situation is defined as invariants. For instance, the distance cannot be changed by Euclidean transformation but is not similarity invariant. The main transformations are as follows:

- Isometries
- Similarity
- Affine
- Projective

Under some projective transformations, the ideal points can be mapped to finite points. However, if the transformation is affine, then it cannot be mapped to a finite point. So, the line $l_\infty$ is fixed if and only if $H$ is affine.

To represent a point in 3D world space, vector of 4 is used. The point $x = (x_1, x_2, x_3, x_4)^T$ represents the vector in homogeneous form. The inhomogeneous form is $x = (x_1, x_2, x_3, x_4)^T$.

## 2.3 Camera Geometry

In this section, the description of the simplest camera model is explained. The name of the camera model is the *pinhole camera model*. The Figure 2.1 shows the modeling of the pinhole camera geometry.

As seen in Figure 2.1, $C$ is the camera center and is the center of the coordinate system. The distance between the image plane and the camera center $C$ is $f$ and it is called *focal length*. The intersection of image plane and the z-axis is $p$ and it is called *principal point*. The projection can simply written like below formula if all the points in both world and image planes represented by homogeneous coordinates.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{2.4}$$

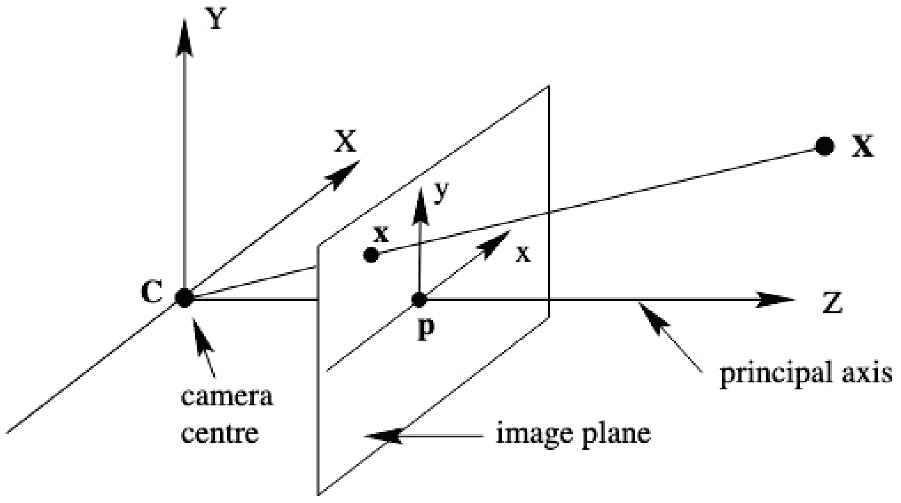


Figure 2.1: Pinhole Camera Model (Source:[10])

In order to make all projections, a projection matrix is necessary. This matrix is notated as $P$ and it is called the *camera matrix*.

$$P = \begin{bmatrix} f_x & s & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.5}$$

There are some properties and definitions of the matrix . The *aspect ratio* is the ratio $f_y/f_x.s$ is *skew* and defines the tilt of the image plane. To calculate the skew, the equation $s = \cot(\alpha)f_y$ can be used.$\alpha$ is the skew angle. The precision of the camera is the key to define the skew and $\alpha$. Mostly, they are assumed 0° and 90°, respectively. The reason behind this is that the cameras are accurate and precise in today's world. And finally, $u_0$ and $v_0$. These two values hold the shift in the coordinate system.

The camera systems work in their own coordinate system. The camera coordinate system is also called Euclidean coordinate frame. However, in order to make calculations and projections, the conversion from Euclidean coordinate system to the world coordinate system must be done. World coordinate system is a system that consists of 3D derivations. These two coordinate systems are related to translation and rotation. To demonstrate the transformation from one to another, let's assume that is a point in the world coordinate system and $X_C$ is the exact same point in the camera coordinate system. Then, the following appears.

$$X_C = R(X - C) \tag{2.6}$$

$C$ represents the center of the camera in the world coordinate system.$R$ is the rotation matrix.$R$ holds the orientation of the camera and it is a $3x3$ matrix. Equation 2.6 is in inhomogeneous form. The homogeneous form is as follows.

$$X_C = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} X \tag{2.7}$$

Equation 2.7 gives us the formula below.

$$x = KR[I \mid -C]X \tag{2.8}$$

where the point is in now world coordinate system. Equation 2.8 can be divided into two parts. The parameters that form $K$ are known as *intrinsic* parameters. These parameters indicate the internal specifications of the camera.$R$ and $C$ consist of *extrinsic* parameters. Extrinsic parameters hold the camera orientation inside the world coordinate system. To write the camera matrix more simpler:

$$x = K[R \mid t]$$

## 2.4 Two View Geometry

In this section, the epipolar geometry is discussed. The epipolar geometry is the relation between two views. It does not depend on the structure of the scene. It is dependent on the internal camera parameters. After describing epipolar geometry, the derivation of the fundamental matrix $F$ can be understood and it is also discussed in this section.

In epipolar geometry, there are two views in 3D space. They are related to a point $X$. The point $X$ is imaged in both views with two different cameras.

As seen in Figure 2.2, there is a point $X$ in the 3D world space and its corresponding

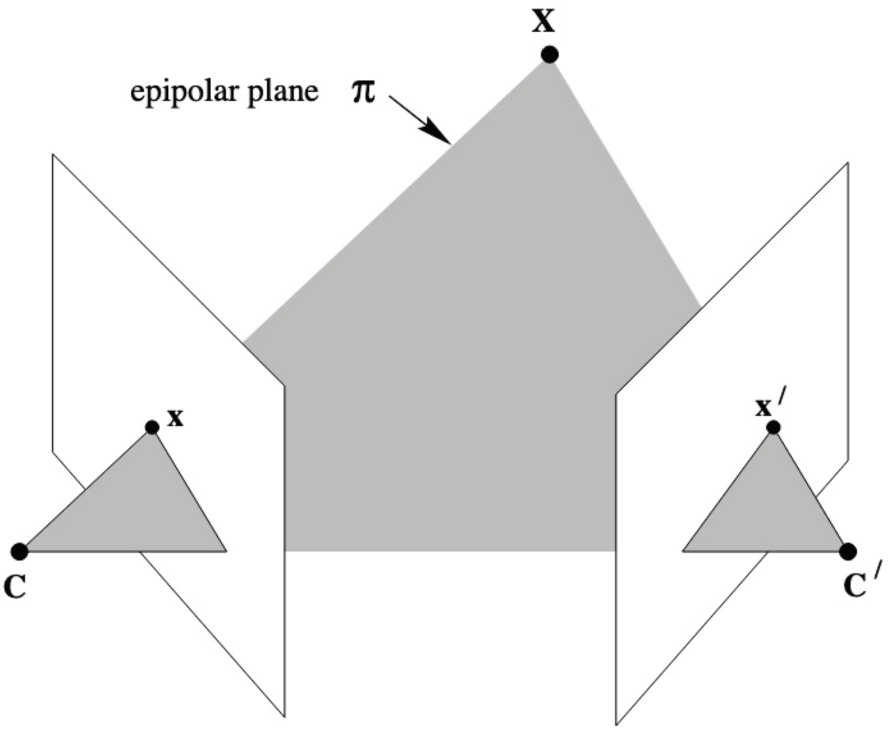


Figure 2.2: Correspondence Geometry (Source:[10])

images are $x$ and $x'$. The camera centers are $C$ and $C'$. All of the objects are located in a common plane and it is called *epipolar plane* $\Pi$.

To understand the epipolar geometry, first, the terminology should be described. As mentioned before, there are two different cameras located in the same 3D plane and two different image planes that intersects with the epipolar plane $\Pi$. The points $e$ and $e'$ are called *epipoles*. The line connecting the camera centers is known as *baseline*. The intersection of the baseline with image planes creates epipoles. The plane that consists baseline is called epipolar plane. And finally, the lines $l$ and $l'$ are called *epipolar lines*. Epipolar line is the line that appears with the intersection of an epipolar plane and an image plane.

As mentioned before, the images of the point $X$ are $x$ and $x'$. There is a 2D mapping between imaged points. The imaged points are projectively equivalent and the mapping between them is a 2D homography $H_\pi$. The homography can be written as

$$x^{'} = H_\pi x$$

To derive the line passes through the point $x'$ and the epipole $e'$

$$l^{'} = [e^{'}] \times H_\pi x \tag{2.9}$$

Where $[e^{'}]_{\times}$ is the skew-symmetric matrix. The fundamental matrix $F$ can be written as $F = [e^{'}] \times H_{\pi}$. The $H_{\pi}$ is the mapping from one imaged point to another imaged point.

There are more than one ways to derive fundamental matrix $F$. In this section, the derivation from the correspondence condition is presented. As mentioned Equation 2.9, the equation of the line $l'$ can be calculated using epipoles and homography. The epipolar line equation can also be written like

$$l^{'} = Fx$$

As mentioned in Section 2.2, if a point lies on a line, the multiplication should be equal to zero. Hence, this is a known fact that the imaged point $x'$ lies on the epipolar line $l'$. And according to the rule, $x^{'T} l^{'} = 0$ must be satisfied. With this equation, the most basic property of the fundamental matrix can be stated. For any pair of correspondence, the fundamental matrix $F$ satisfies the condition

$$x^{'T} F x = 0 \tag{2.10}$$

Equation 2.10 must be satisfied. It is a necessary condition to say that two points are correspondent. This is true because the imaged points $x$ and $x'$ are correspondent and $x'$ lies on the epipolar line $l'$.

To summarize, the most essential properties of the fundamental matrix are as follows.

- The fundamental matrix has 7 degrees of freedom.
- $F$ has a rank of 2.
- $x^{'T} F x = 0$
- $l^{'} = Fx$ and $l = F^{T} x^{'}$
- $Fe = 0$ and $F^{T} e^{'} = 0$

To reconstruct the cameras and the planar structure from two view, three main steps can be followed. Calculating the fundamental matrix from the known point correspondences. The camera matrices can be derived from the fundamental matrix and the next step is calculating the camera matrices. And lastly, calculation of all point correspondences in the space. In order to do these steps, one needs to compute fundamental matrix. There is a minimum number of point correspondences to compute the fundamental matrix. At least 8 point correspondences are necessary to solve linearly for the fundamental matrix.

However, there is a nonlinear solution that uses only 7 point correspondences. The 8-point version is the general way of calculating the fundamental matrix $F$.

The fundamental matrix is given by the equation

$$x'^{T}Fx = 0$$

Each point correspondence gives one linear equation. These equations are the unknown entries of the fundamental matrix. Let's assume that the points are $(x, y, 1)$ and $(x', y', 1)$. The entries of the equation can be written as

$$x'^{T}Fx = \begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.11}$$

Then, Equation 2.11 becomes

$$= x'xf_1 + x'yf_2 + x'f_3 + xy'f_4 + yy'f_5 + y'f_6 + xf_7 + yf_8 + f_9 \tag{2.12}$$

$f$ values are the entries of $F$ in row-major order. Next, Equation 2.12 may be expressed as vector inner product.

$$\begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} f = 0 \tag{2.13}$$

There is a set of point correspondences and to get linear equations

$$Af = \begin{bmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & y_n' & x_n & y_n & 1 \end{bmatrix} f = 0 \tag{2.14}$$

Equation 2.14 is a homogeneous solution. To solve this equation, matrix $A$ should be at a rank of no more than 8. If the rank is 8, the solution can be defined as unique and using linear methods is enough to find the answer.

There can be noise in the point coordinates. If that is the situation, the rank of $A$ can be greater than 8. This case brings us to a least-squares solution. The smallest singular value of $A$ is the solution for $f$. The smallest singular value is the last column of $V$ in the singular value decomposition (SVD) of $A$. The solution can be written in the form

$$SVDA = UDV^T$$

In order to find a unique solution, the choice of a constraint is needed. To find the $f$, the condition is $||f|| = 1$. The solution is the vector that minimized the $||Af||$ subject to $||f|| = 1$. The algorithm described is the *8-point algorithm*.

## 2.5 Triangulation

This section describes how to calculate the position of a point in the world coordinate system by using its image coordinates in image planes and the camera matrices of both two views. Naturally, there may be an error while back-projecting the point. The best solution needs an optimal cost function. In order to find a cost function, an invariant triangulation method is necessary. It should be invariant to projective transformations of 3D space.

As mentioned above, there are errors while back-projection points. Back-projected points can be skew if the measured points have errors. If there are errors, $x = PX$ and $x^{'} = P^{'}X$ cannot be satisfied. Also, the epipolar constraint $x^{'T}FX = 0$ cannot be satisfied by the image points.

The optimal triangulation method must be invariant to reconstruction transformations. To denote a triangulation method to find a 3D world point

$$X = \tau(x, x^{'}, P, P^{'}) \tag{2.15}$$

As written in Equation 2.15, the triangulation must be invariant to transformation $H$. Hence,

$$\tau(x, x', P, P') = H^{-1}\tau(x, x', PH^{-1}, P'H^{-1})$$

This indicates that the triangulation formula uses the transformed camera matrices in the transformed point coordinates.

In linear triangulation method, the DLT similar way is used. There are measurements that are already known, $x = PX$ and $x' = P'X$. These equations can be merged into $AX = 0$. For the first image,

$$x(p^{3T}X) - (p^{1T}X) = 0$$

$$y(p^{3T}X) - (p^{2T}X) = 0$$

$$x(p^{2T}X) - y(p^{1T}X) = 0$$

where the elements notated as $p^iT$ are the row elements of $P$. These are the linear equations.

Therefore, equation $AX = 0$ can be used.

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \tag{2.16}$$

Two equations from two images are combined in Equation 2.16. This gives us a four homogeneous unknowns. This can be solved with DLT.

## 2.6 Summary

In this chapter, background information on projective geometry, camera geometry, two view geometry and triangulation are presented. Some of the well-known algorithms are also presented like 8-point algorithm.

As seen in Section 2.2, the projective geometry builds the ground floor of all the topics that will be covered in this thesis. Equations and derivations that are used in this section are used largely in the following parts.

In Section 2.3, the camera geometry is described. The simple but widely used pinhole camera is explained. Understanding the structure of the camera is necessary to comprehend the calibration concepts. The parameters like focal length and principal point are explained. Two subdivisions of the camera matrix, intrinsic and extrinsic, are described. The derivation of the matrix $K$ is also described.

In Section 2.4, the most essential concepts in two view geometry are described. Two view geometry is an important topic while dealing with camera calibration. The fundamental matrix $F$ is used in almost every calibration method and the derivation of the $F$ is explained in this section. In order to find point correspondences in two image planes and their related 3D world coordinate, the fundamental matrix is an important device.

And at the end in Section 2.5, the linear triangulation method is described. In order to test the accuracy of the projections, the triangulation methods are necessary.

# CHAPTER 3

# CALIBRATION TOOLS

## 3.1 Introduction

In this chapter, the main tools of this work are explained. These tools are two calibration tools from different published works. The core calibration method in both calibration tools is described. This method is not a new method but the impact is huge and it is still being used. The method is Zhang's method [30] from OpenCV to other libraries, it is implemented. Two calibration tools are compared and verified with the proposed evaluation method in the next chapter.

Section 3.2 describes the most used calibration method. The name of the method is "A Flexible New Technique for Camera Calibration.". The work is published in 2000 by Zhang. As it is written in the paper, it describes a flexible way of camera calibration. The reason for the name flexible is that this method can be applied without prior specialized knowledge of computer vision or two view geometry. The only requirement is at least two planar views in different orientations. The work also consists of radial distortion.

Section 3.3 explains the work called "Efficient Pose Selection for Interactive Camera Calibration.". This paper is published in 2019. This method is one of the works compared in this thesis. The planar pattern choice is important in camera calibration yet this is barely debated. This work proposes a set of calibration poses for interactive calibration. The main distinction of this work is the consideration of different planar orientations and numerical values of positions. Further information is in the section.

Section 3.4 explains the other calibration tool. The name is "AprilCal: Assisted and repeatable camera calibration.". Tool uses the current state of the calibration to suggest the next pose. A simultaneous advising system is the main difference of this work. As pattern, AprilTags [20] are used. The other contributions of this work are two new score functions. One is for focal length accuracy and the other is for pixel uncertainty. Further information is in the section.

## 3.2 Calibration Method

The method that is presented in this section is the base method for both of the calibration tools. The core calibration method is the Zhang's [30] calibration method. After gathering the necessary and required poses, tools are executing this calibration method to calibrate the camera.

The only requirement for this method is having a camera and a planar pattern. This is why it is called flexible at first. The condition is having at least two frames with different orientations. The motion of the planar surface can be anything. It can be moved by hand or a device. This method is in both the fields of photogrammetry and self-calibration. The reason of this is the method uses 2D information.

The main assumption of this method is that the model plane is on $Z = 0$ of the world coordinate system. To show the homograph between the model and its image plane

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$= A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Since there is an assumption over the axis $Z$, the point can be expressed as $(X, Y)^T$. Hence, the relation between the point in the world coordinate system and its imaged one can be described by a homography.

$$sx = HX \tag{3.1}$$

$$H = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{3.2}$$

In equations 3.1 and 3.2, $H$ is a $3x3$ matrix and $s$ is the scale factor.

The process finishes in three main steps. It starts with an analytical solution, then continues with nonlinear optimization. This optimization is based on the maximum likelihood. And for the final step, the addition of lens distortion is done.

For the first step, there is a closed-form solution. Let

$$B = A^{-T}A^{-1}$$

$$\text{gives} \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}$$

where $B$ is a symmetric matrix. Then, $b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$. By adding the homography, the equation becomes

$$h_i{}^T B h_j = v_{ij}{}^T b$$

where $h_i$ means the $i^{th}$ column vector of the homography H. And $v_{ij}$ consists of the homograph vectors like

$$v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$$

Furthermore, the two fundamental equations can be written as homogeneous equations.

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0$$

Let's say that there are $n$ images in the set. This gives us $n$ equations. Stacking these equations

$$Vb = 0$$

The equation above gives us different solutions with the number of $n$ images. If $n \geq 3$, the solution is a unique solution $b$ and it is constrained by a scale factor. If $n = 2$, equation becomes $[0, 1, 0, 0, 0, 0]b = 0$. If $n = 1$, there is a solution that only gives two camera intrinsic parameters. To estimate general $b$, finding the right singular value vector of $V$ related with the smallest singular value gives the solution. After estimating $b$, all intrinsic parameters can be computed.

After getting the camera intrinsic matrix, computation of the extrinsic parameters can be done easily.

$$r_1 = \lambda A^{-1} h_1$$

$$r_2 = \lambda A^{-1} h_2$$

$$r_3 = r_1 \times r_2$$

$$t = \lambda A^{-1} h_3$$

where $\lambda = 1/||A^{-1}h_1|| = 1/||A^{-1}h_2||$

The next step is refining the solution above with a maximum likelihood estimation. This needs to be done because the equation above is calculated with minimizing an algebraic distance.

Let's say that there are $n$ images and points in the world coordinate system. The maximum likelihood estimation can be done by the following function.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} ||m_{ij} - m(A, R_i, t_i, M_j)||^2$$

where $m(A, R_i, t_i, M_j)$ is the projection of the points. In order to minimize, Levenberg-Marquardt[14][17] is used.

Until this step, the camera was considered a pure camera without lens distortion. However, cameras have lens distortion and it is especially radial distortion. In this work, only two values of radial distortion is described.

Let $(u, v)$ be the distortion-free image coordinates and $(\tilde{u}, \tilde{v})$ the related real image coordinates. Also, $(x, y)$ and $(\tilde{x}, \tilde{y})$ are the coordinates with the same properties above, respectively. Hence,

$$\tilde{x} = x + x[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

$$\tilde{y} = y + y[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

where $k_1$ and $k_2$ radial distortion coefficients. Assuming $\gamma = 0$ gives

$$\tilde{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

$$\tilde{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

## 3.3 First Calibration Algorithm

There have been various works in the camera calibration tool field. Most of them used different methods to relax the user. Naturally, user dependent calibration is error prone. In this work, the choice of planar pattern poses is distinctive. A robust and compact set of calibration poses is proposed. The generation of optimal pattern poses considers avoiding degenerate pattern compositions. For each individual camera parameter, a relation between pose and constraint is described. This reduces the generation time compared to other work [23] that is presented in this thesis.

The covariance of the parameter solution is used during the calibration. The uncertainties gradually are reduced step by step with the generated poses. The main contributions of this work are:

- Proving the need for two different pose generation strategies
- Including both strategies into an efficient pose selection algorithm

In order to analyze the error, the variance of the estimated parameters is used. Index of dispersion is used to correlate the parameters to each other. The proposed method works fine with every planar calibration pattern like the chessboard pattern. However, ChArUco [7] marked pattern is used in this work. The reason for this is fast detection. Real-time calibration requires fast detection of the planar pattern. The size of the pattern is $9x6$.

In order to calibrate the camera, Zhang's [30] method is used. Selection of the poses depends on the different key frames. While choosing frames, they proposed to split the parameter group into pinhole and distortion parameters. In the estimation process, some ambiguities appear. One is between the focal length and the distance to the camera and the other is between translation and principal point. Both of these ambiguities can be solved by tilting the pattern. For distortion parameters, accurate measurement in the image regions is required.

After stating the parameter splitting, the splitted parameters are as follows.

$$C_K = [f_x, f_y, c_x, c_y], C_\Delta = [k_1, k_2, k_3, p_1, p_2]$$

Regarding the work, avoidance of singular poses is a must. In order to get rid of pinhole singularities, there are three configurations. These configurations are as follows:

- Image plane and pattern must not be parallel to each other.
- Each image axes must not be parallel to the pattern.
- The vanishing lines of the planes must not be reflections of each other.

Satisfying these conditions helps the calibration process and each step adds information.

The pose generation process of this work is described in 4 steps. The purpose of these steps is to maximize the angular spread between the pattern and the image plane. These steps are:

1. Choose a distance that is far enough to capture the whole pattern.

2. Tilt the pattern in the range of $-70°$ and $70°$. This range is calculated using the interpolation of binary subdivision of $(0, 1)$. This step maximizes the angular spread.

3. Rotate the view by $22.5°$. This prevents the pose to be parallel to one of the axes.

4. Shift every image axis by 5% of the image size. This step is for faster convergence and increases the spread for each axis.

The steps above are for the intrinsic parameters $C_K$. For distortion parameters $C_\Delta$, increasing sampling accuracy reduces distortions. To do this, the generation of a distortion map is done. Search for the distorted parts is done in 4 steps and as follows:

1. Threshold the map to find the highest distortion.
2. Fit a bounding box to the distorted region.
3. Exclude that region while searching.
4. Align the pattern according to the top-left corner of the bounding box.

To initialize the camera calibration and get the first calibration parameters, the calibration method [30] requires at least two frames. The selection of poses for the first two frames is as follows.

- For the $C_K$, tilt the pose by $45^\circ$ around the axis x.
- For $C_\Delta$, get a pose that is parallel to the image plane.

After getting the initial frames, the calibration process starts. The main goal is to minimize variance of the parameters. This process works with one parameter each time. The choice is determined with the help of the index of dispersion. The calibration process is done after the convergence. For determining convergence, a ratio test over parameter variance is done. If the refinement is below the given threshold, there is no need for the next step. The calibration ends after all parameters have converged.

In this tool, there is a user guidance system like the other calibration tools. The system guides the user until the desired number of poses is reached. In order to detect a sufficient number of poses, Jaccard index is used. This is computed with target pose $T$ and estimate pose $J(T, E) > 0.8$. If , the desired number of poses is reached.

The results are computed over 5 iterations. The method described in this section requires 70% of the images required by AprilCal [23].

## 3.4 Second Calibration Algorithm

In this section, the second calibration tool which is called AprilCal is described. Most of the other tools stores some predefined poses and asks the user to move the pattern to these poses. After gathering sufficient poses, calibration starts. However, in this work, the next is generated with the help of the previous pose. Parameters are checked after each step to satisfy the user requirements.

To suggest the next pose, reliable score functions are necessary. This work provides two methods for scoring. One is for focal length estimation and the other is for the model uncertainty.

The main contributions of this work are listed below.

- Suggestion framework for generating poses during camera calibration.
- New score functions to increase the accuracy of the calibration.
- A new bootstrapping method to start calibration.

The system works with AprilTags [20]. The reason is that these tags are easy to detect and more robust. Even if some parts of the pattern are not visible, AprilTag can be detected. The system suggests new poses and the user moves the pattern on that position. Then, the next pose is computed regarding the previous pose. This process continues until the desired accuracy that the user defines is reached.

The calibration process is divided into two parts, bootstrapping and uncertainty reduction. In bootstrapping phase, new images are added to determine the first camera parameters. After the calculation of the first parameters, the system continues with the uncertainty reduction phase. In bootstrapping phase, quick initialization is necessary. So, focal center is assumed at the center of the image. This gives the focal length estimate after the first frame. In order to select the next best pose with less accurate parameters, the selection is done using the variance. Intrinsic parameters with a low variance are desired. After each step, the previous inaccurate frame is replaced with the new lower intrinsic uncertainty.

After constraining intrinsic parameters, the next step is increasing the accuracy of the calibration. In order to achieve this, “Max Expected Reprojection Error” is proposed. It can be calculated in every step by sampling the current distribution over parameters. The reprojection error is a geometric distance error corresponding to the distance between a projected point and a real one. It is used to quantify how closely an estimate of a 3D point is to its projection.

Table 3.1: AprilCal Mean and Max Errors

| **Dataset** | **Lens Model** | **Mean Reproj. Error** | **Max Reproj. Error** |
|---|---|---|---|
| OpenCV | Radial, 3 distortion terms | 0,728 | 38.646 |
| AprilCal | Radial, 3 distortion terms | 0,229 | 1.651 |
| AprilCal | Angular, 4 distortion terms | 0,203 | 1.444 |

In the Table 3.1, it can be seen that increasing distortion terms decreases the reprojection error. Also, the usage of both Mean Reprojection Error and Max Reprojection Error makes camera calibration more robust.

The Max Expected Reprojection Error is described as a calibration quality measure. Users specify their desired accuracy and then if the Max ERE reaches that accuracy, calibration can be called accurate. Their proposed error metric relates to the reprojection errors of the captured test set. To test the accuracy of partial calibration, Max ERE gives high accuracy.

# CHAPTER 4

# EVALUATION TOOL

## 4.1 Introduction

In this chapter, the proposed evaluation system of calibration tools is presented. The evaluation process has three main steps. These steps are as follows.

1. Setup 3D modeling system
2. Generate suggested poses
3. Test generated poses

To gather the suggested poses, given pose generation steps are followed. Then, with the correctness of 3D modeling camera, results are compared.

Section 4.1 describes the camera and system setup. The specifications of the system used in this evaluation tool are presented. The modeling tool used is also described. Its basic functionalities and necessary information are included.

Section 4.2 describes the way of generating recommended poses. In this section, each step of pose generation is explained in detail.

In Section 4.3, the results are compared. With the generated poses, each calibration tool is tested. Knowing the camera ground truth values, calibration tool outputs are compared.

## 4.2 Evaluation Tool Setup

The camera system used in this thesis is a virtual camera provided in the modeling tool. The camera is a perceptive camera. Perspective camera is the same as how humans see in the real world. Objects in the distance appear smaller. The camera can be manipulated with the controls given in the modeling tool. In Table 4.1, you can see the details of the system.

In order to calibrate the camera with the OpenCV [3], a checkerboard pattern is required. It is a specific pattern and has some metrics. Used pattern is $9x6$ squared. It is provided in OpenCV [3]. The pattern used is as follows.

Table 4.1: System Details

| **System** | **Properties** |
|---|---|
| Macbook Pro 2018, 16 | 2.6GHz 6-core Intel Core i7 |
| Camera | 720p FaceTime HD camera |
| Modeling Tool | Blender |

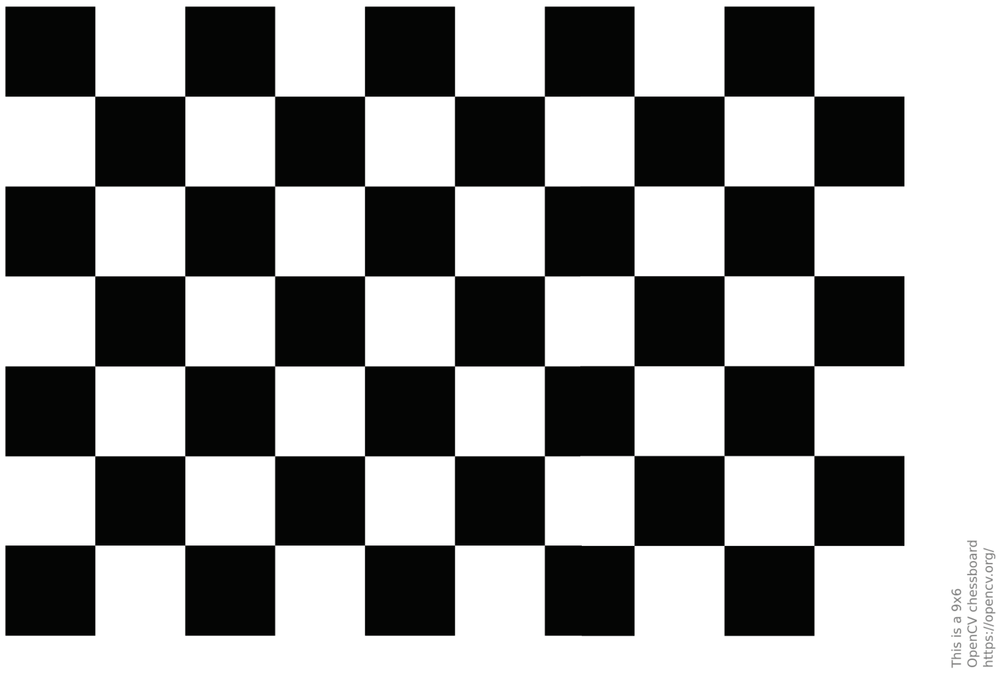


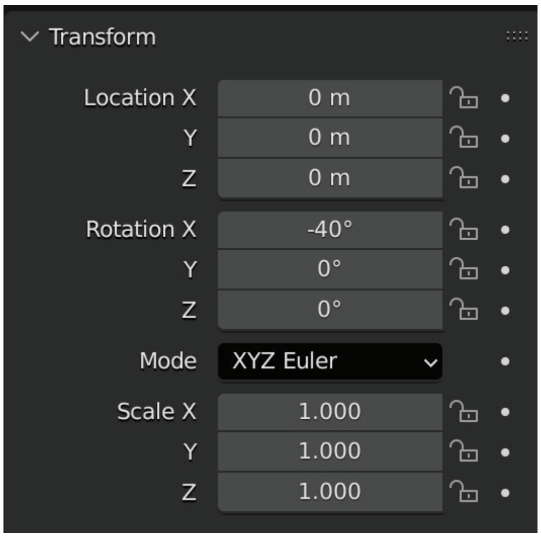
Figure 4.1: Checkerboard Pattern (Source[3])

The 3D modeling tool is named Blender [6]. It is an open-source software that is licensed as GNU GPL and owned by its contributors. It has a big array of modeling tools. With the help of this, modeling and transforming is very easy. The main reasons to use this tool are:

- Sculpting tools
- High resolution subdivision
- Python scripting

Transformation settings are location, rotation and scale. In order to change them, typing the necessary values are enough.

Values in Figure 4.2 are the same as used those in projective geometry. All of them are in 3D world space.


Transform

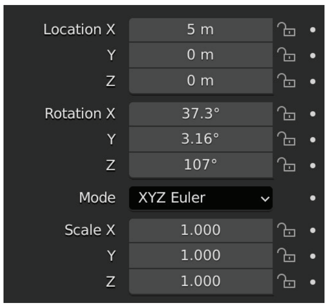
| | |
|---|---|
| Location X | 0 m |
| Y | 0 m |
| Z | 0 m |
| Rotation X | -40° |
| Y | 0° |
| Z | 0° |
| Mode | XYZ Euler |
| Scale X | 1.000 |
| Y | 1.000 |
| Z | 1.000 |

Figure 4.2: Transformation Values From Blender

## 4.3 Pose Generation

Each calibration tool has its own steps to generate poses. The purpose of the proposed method is to imitate these steps in the modeling tool and generate the required poses. Every step is well-defined in calibration tools.

First step of pose imitation is setting up the scene in the modeling tool. There are three main objects in a simple scene. These are camera, light and object. The object is a checkerboard pattern in this method. The light is placed on top of every object in the scene. The reason for this is lighting every object clearly to be seen by camera. The coordinates and transformations of the light are as follows:



| | |
|---|---|
| Location X | 5 m |
| Y | 0 m |
| Z | 0 m |
| Rotation X | 37.3° |
| Y | 3.16° |
| Z | 107° |
| Mode | XYZ Euler |
| Scale X | 1.000 |
| Y | 1.000 |
| Z | 1.000 |

Figure 4.3: Position and transformation of the light taken from blender

The other object is camera. It is a perspective camera. Like every other real camera, it also has $K$ and $P$ matrices. These matrices are given below, respectively.

$$\begin{bmatrix} 2666.67 & 0.00 & 960.00 \\ 0.00 & 2666.67 & 540.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix} \text{and} \begin{bmatrix} -960.00 & 2666.67 & 0.00 & 4800.00 \\ -540.00 & 0.00 & -2666.67 & 2700.00 \\ -1.00 & 0.00 & 0.00 & 5.00 \end{bmatrix}$$

The matrices above are ground truth data. While comparing calibration tools, these values are necessary. Figure 4.4 shows the described scene visually.

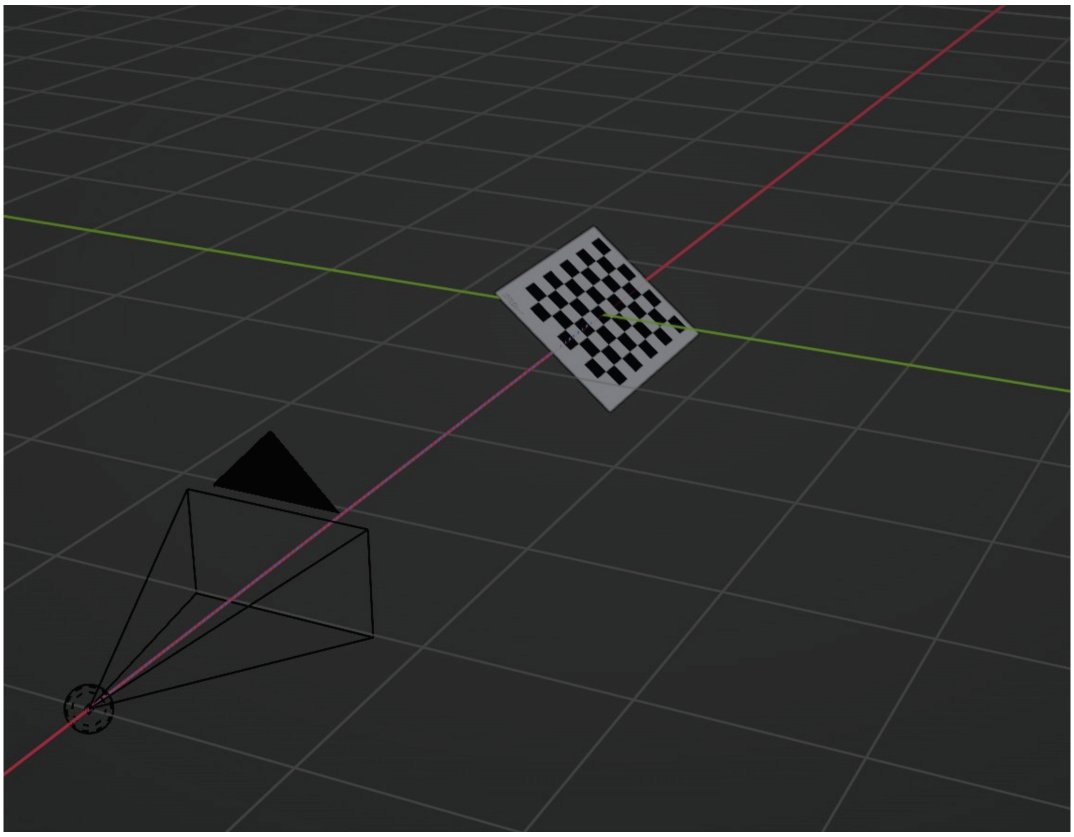
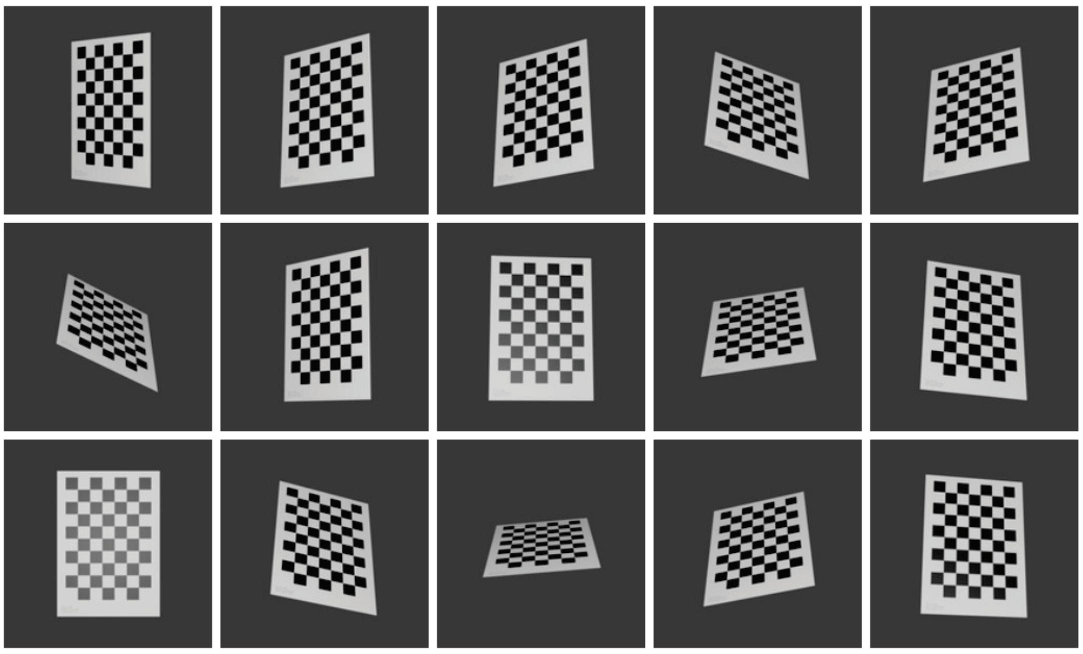
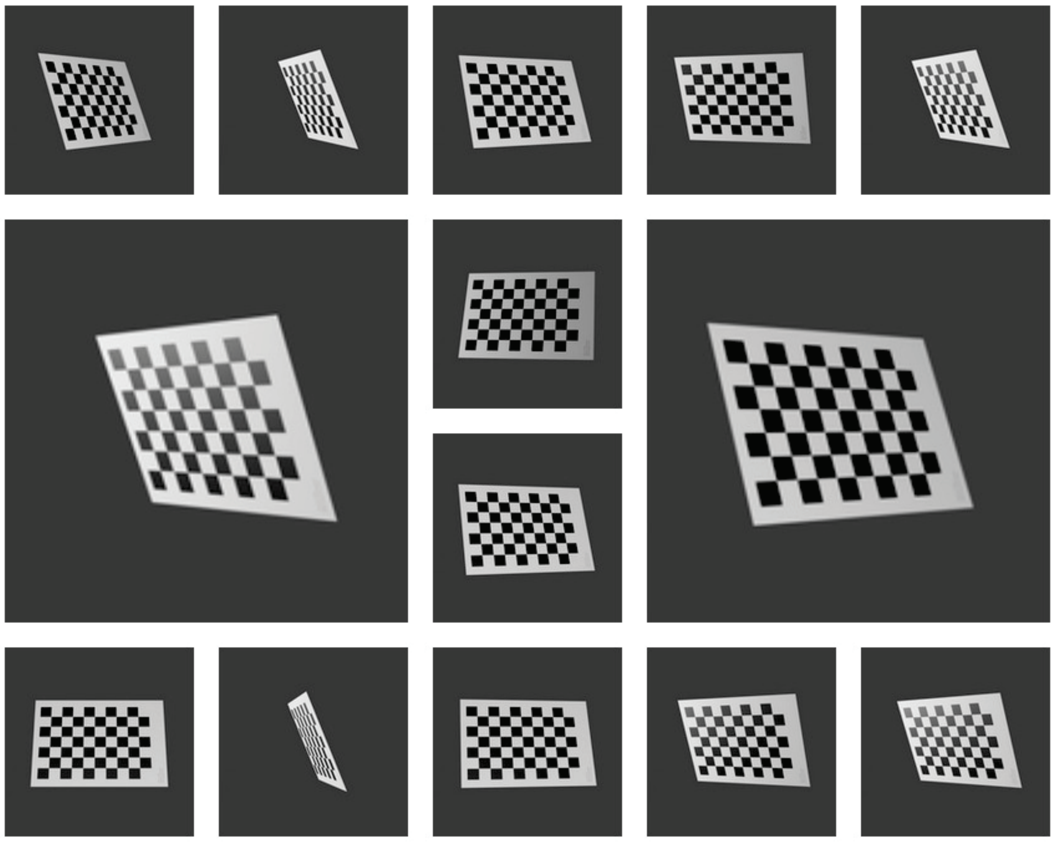


Figure 4.4: Scene taken from Blender

The next step is generating poses. All algorithms have different stopping metrics. Poses are generated with respect to these metrics. These metrics are mentioned in the previous chapter. For AprilCal, 15 frames are generated. For PoseCalib, 14 frames are generated. The generated frames for each calibration tool can be seen in Figure 4.5a and Figure 4.5b.



(a)



(b)

Figure 4.5: (a) AprilCal (b) Pose Selection

After generating the poses presented in the Figure 4.5a and Figure 4.5b, the next step is feeding OpenCV [3] calibration function with these images. As mentioned before, these poses are generated with each proposed approach. The reason to use OpenCV [3] calibration function is that. Both methods are using Zhang's [30] method in their work.

## 4.4 Results

Both calibration methods are explained and their pose generation steps are applied. Suggested poses used in calibration function and calibration results are gathered. In this section, all results are compared.

Table 4.2: Generated poses and errors

| Method | Frames Used | Mean Reproj. Error |
|---|---|---|
| AprilCal | 15 | 0.005 |
| Pose Selection | 14 | 0.015 |

Table 4.2 shows that AprilCal gives better results without considering the number of images. The difference is small and Pose Selection method requires fewer images.

Using a modeling tool gives the opportunity to compare camera matrices with a ground truth camera matrix. The camera matrix results of both calibration tools can be seen below and can be compared with the ground truth matrix. The first matrix is ground truth. Others are AprilCal and Pose Selection, respectively.

$$\begin{bmatrix} 2666.67 & 0.00 & 960.00 \\ 0.00 & 2666.67 & 540.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

$$\begin{bmatrix} 2686.06 & 0.00 & 950.68 \\ 0.00 & 2685.68 & 530.75 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

$$\begin{bmatrix} 2668.64 & 0.00 & 915.92 \\ 0.00 & 2672.31 & 671.07 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

AprilCal's focal lengths are much higher than the ground truth. The reason for this is AprilCal's tries to create intrinsic parameters as faster as it can. The results above show us that Pose Selection method estimates focal lengths better than AprilCal but the principal points are estimated more accurately compared to Pose Selection.

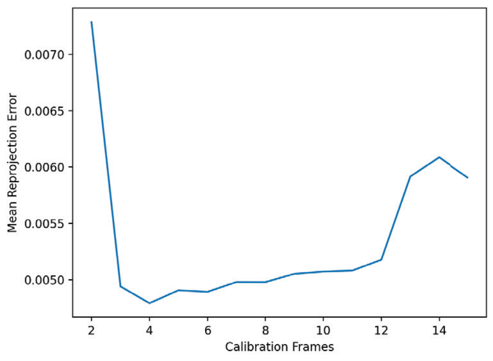


Figure 4.6: AprilCal Mean Reprojection Error

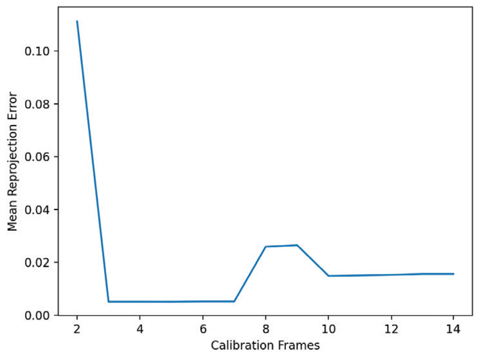


Figure 4.7: Pose Selection Mean Reprojection Error

Figure 4.6 and Figure 4.7 show that AprilCal is better than Pose Selection with fewer images. We can say that both of the algorithms suggest good frames to increase the calibration quality. AprilCal's first suggested frames are better than Pose Selection because starting mean reprojrojection errors are lower than other algorithm. The other thing to say is Pose Selection suggests better poses because error decreases much more than AprilCal.

Figure 4.8 and 4.9 show that both focal lengths converge and they are almost the same. Pose Selection is slighty better.

Figures 4.10, 4.11, 4.12 and 4.13 show us that AprilCal is better at finding principal points and also more stable and requires fewer images. For distortion parameters, Pose Selection is more stable but AprilCal is more accurate.

For an evaluation tool, one of the most necessary requirements is to create repeatable outputs and do tests. For a calibration process, taking ten pictures and doing tests is

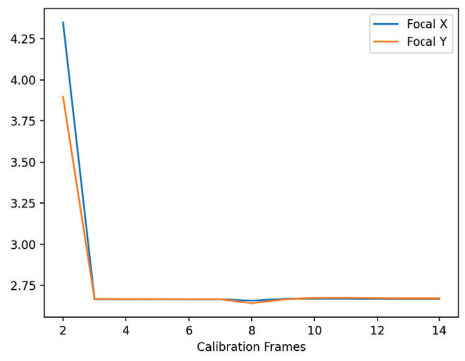


Figure 4.8: Focal lengths change for AprilCal over images

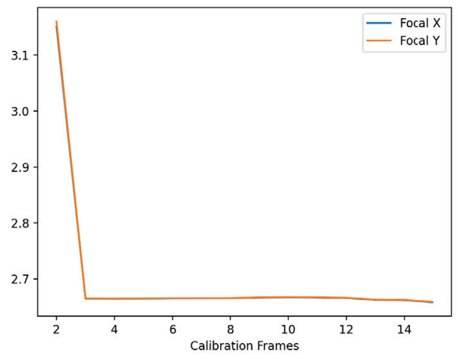


Figure 4.9: Focal lengths change for Pose Selection over images

humanly work. However, doing hundreds of tests is harder and more exhaustive. With this evaluation tool, one can create hundreds of calibration poses and do repeatable tests without being in front of the computer.

In this proposed evaluation tool, one can easily create dozens of test sets. For this purpose, a set that consists of different poses created with the guidance of the calibration algorithms is created. With the help of a Python script, hundred pose group is created. Every pose in these sets is generated using calibration algorithms and they are not exactly the same. They have some similar poses but overall they are not the same.

For ApriCal, each set has 15 calibration frames generated in its way. For Pose Selection, each set has 14 calibration frames and these are also generated with the help of Pose Selection suggestion algorithm. To summarize:

- AprilCal: 100 * 15 Frames
- Pose Selection: 100 * 14 Frames

With this test, the main focus is consistency. A calibration algorithm should be consistent with the changing poses. For each parameter group, results should not change over set changing. In this test case, evaluation tool generates 100 sets of calibration images

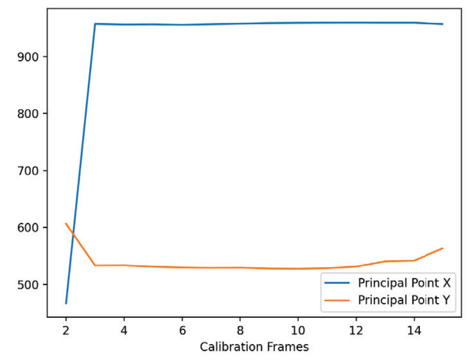


Figure 4.10: Principal point change for AprilCal

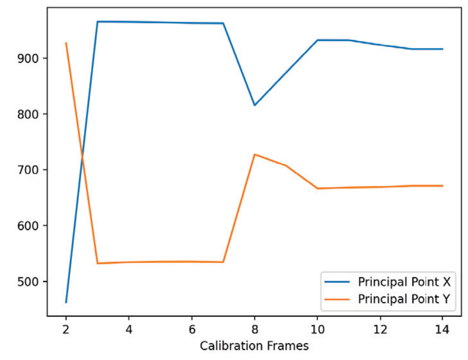


Figure 4.11: Principal point change for Pose Selection

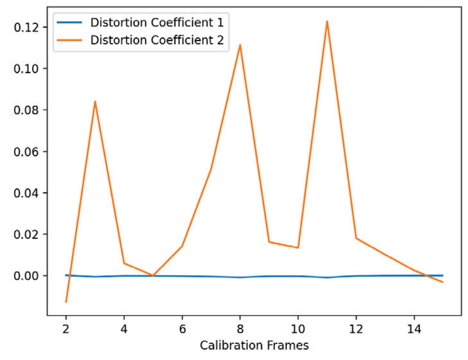


Figure 4.12: Distortion coefficient change for AprilCal

using both algorithms. We are doing this to evaluate the pose generation algorithms suggestion mechanism's stability. For each set, the starting frames are the same but the next frames are generated according to algorithm. As a result, we can see that AprilCal's pose generation mechanism is more stable because the errors are not that much different between sets.

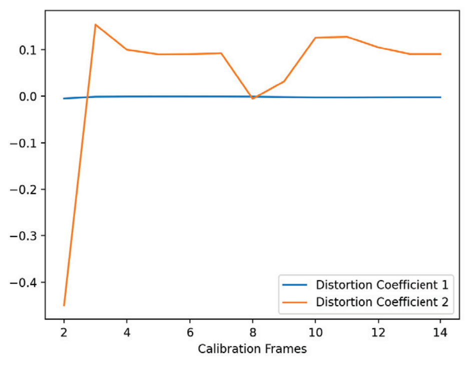


Figure 4.13: Distortion coefficient change for Pose Selection

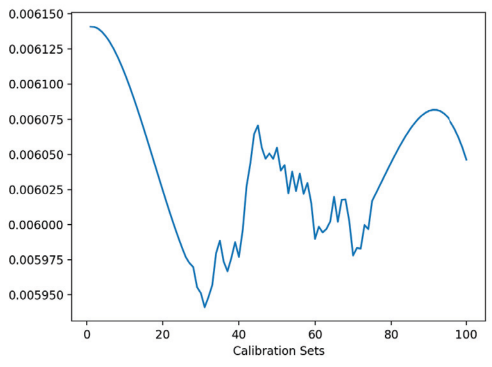


Figure 4.14: AprilCal MRE change over 100 sets

As can be seen in Figures 4.14 to 4.19, AprilCal is more consistent than Pose Selection. MRE changes are much lower and this indicates that AprilCal generates more similar poses. Like MRE changes, focal point changes are also much lower in AprilCal. This also shows that AprilCal can calculate more accurate calibration matrices rather than Pose Selection. At last, in contrast to other metrics, principal point changes over sets are similar in both calibration algorithms.

The next metric is to use triangulation method to determine the 3D point locations in the scene. Triangulation is the method to find the 3D coordinate of a point in its two or more image projections. When the coordinates of the pointing two projections are known, the two rays in space can easily be calculated. The triangulation is the method that finds the ray intersection between them [2]. After finding the calibration matrices with the given algorithms, the next step is to obtain projection matrices. Multiplying camera matrices with the rotation and translation matrices gives the projection matrices. The rotation and translation matrices are obtained from the modeling tool. With the help of the script, the ground truth rotation and translation matrices can be calculated.

For the triangulation projections, there are two images from the same scene generated with the modeling tool. In this scene, some ordinary objects are placed. To continue the triangulation process, a point is located and its coordinates in two image projections

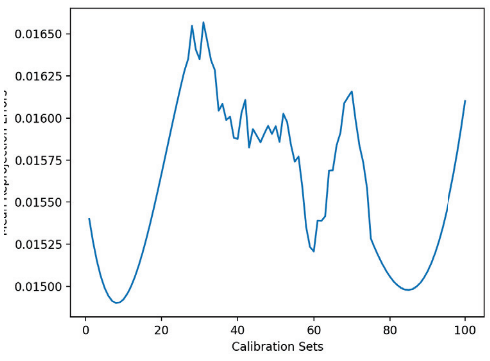


Figure 4.15: Pose Selection MRE change over 100 sets

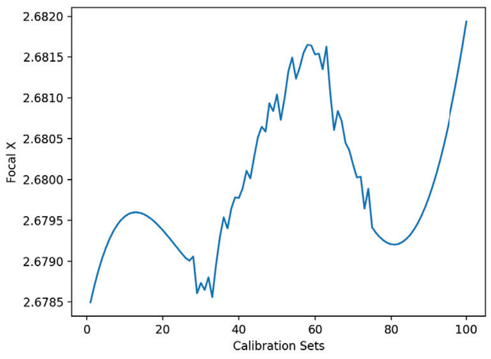


Figure 4.16: AprilCal focal X change over 100 sets

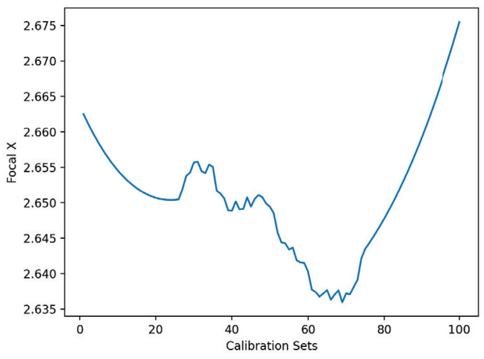


Figure 4.17: Pose Selection focal X change over 100 sets

are obtained. To find the same location, Scale Invariant Feature Transform (SIFT) is used. SIFT algorithm is used to locate the same points in the same scene but with different views. Some algorithms are dependent on the viewpoint, depth and scale but SIFT is independent [16]. Two projection views and the located point can be seen in Figure 4.20.

In the previous steps, the camera matrices are calculated. By using these matrices, the final step can be achieved. The location of the specified point is known as ground truth by the modeling tool. It is a 3D point in the world coordinate system. With the triangulation, the coordinates can be found by using the previously calculated camera

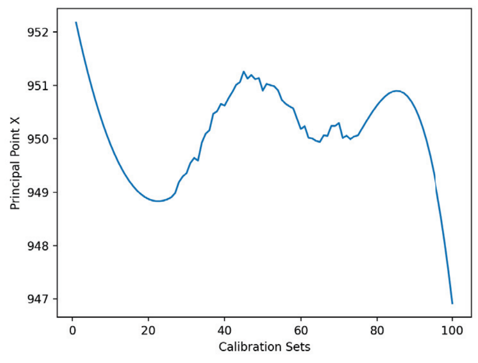


Figure 4.18: AprilCal principal point X change over 100 sets

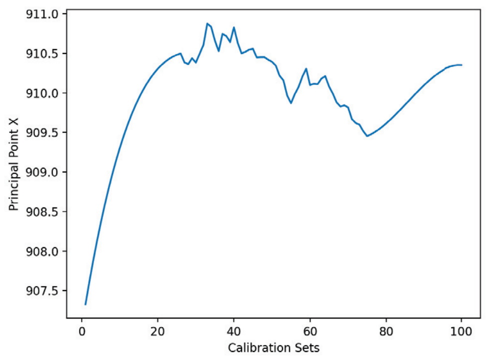


Figure 4.19: Pose Selection principal point X change over 100 sets

matrices with the tested algorithms. The ground truth coordinates are:

- X: 0.2146
- Y: -0.8463
- Z: 0.0435

In order to do triangulation, the projection matrices need to be calculated as mentioned above. After that, the algorithm called Direct Linear Transform (DLT) is used over key point matches. There are two DLT methods. In this work, 3D DLT algorithm is used. To convert from 2D DLT to 3D DLT, the dimension of the problem needs to be changed. With the DLT reconstruction, it is possible to project 2D points into 3D points. The only requirement is to have two different cameras at least [1].

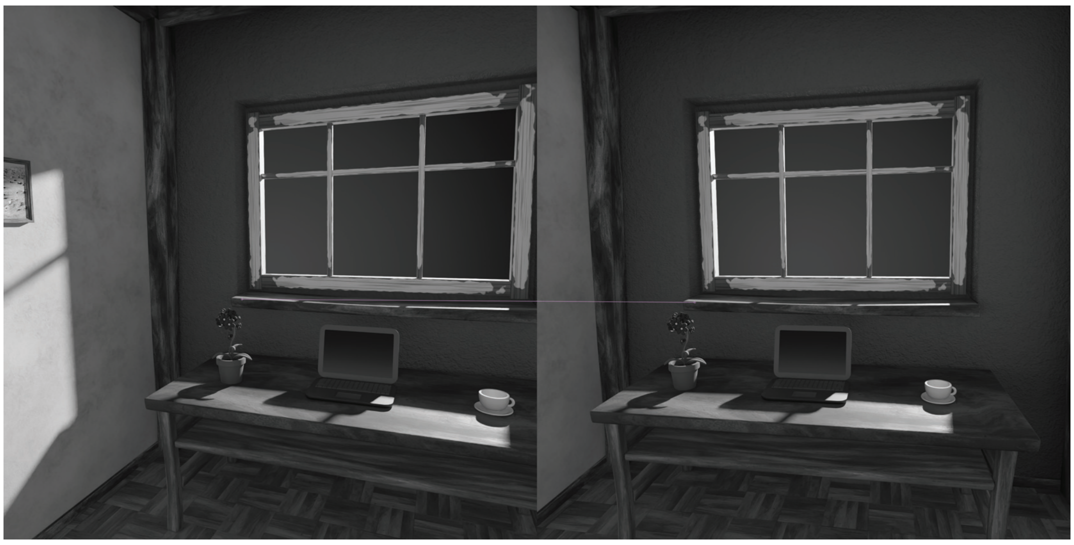


Figure 4.20: Generated scene and the point with Blender

After iterating over points and using DLT, for each camera calibration a 3D coordinate is estimated. For AprilCal:

- X: 0.2147
- Y: -0.8465
- Z: 0.0436

For Pose Selection:

- X: 0.2147
- Y: -0.8464
- Z: 0.0435

The distance between the ground truth and the estimated points can give accuracy. For the distance calculation, Euclidean distance is used. Table 4.3 shows the distances.

Table 4.3: Euclidean distances between the ground truth and estimations

| | **AprilCal** | **Pose Selection** |
|---|---|---|
| Ground Truth | 0.000245 | 0.000141 |

The distance and triangulation calculation above is for only one point in the world system. In order to create a more reliable test metric, more than one point is necessary.

By using key point detection method mentioned above, 329 point matches are found in the same scene between two views. The key points found in the views can be seen in Figure 4.21.

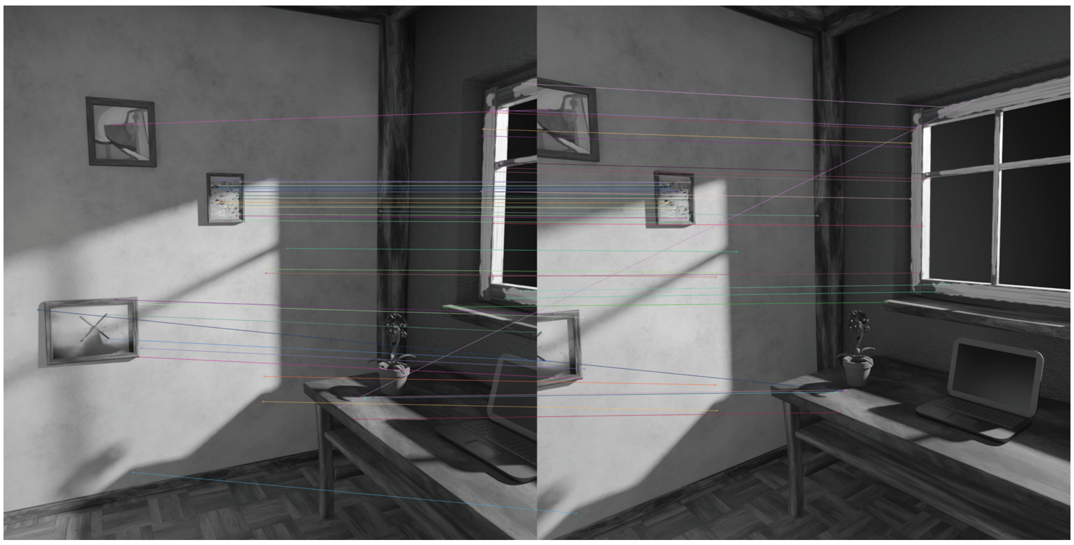


Figure 4.21: Generated scene and the 329 points with Blender

The ground truth coordinates are obtained for all the correct matches using the modeling tool. Using these correct point coordinates, triangulation process can be started. 3D coordinates of each matched point are estimated after iterating over 329 points and using the DLT algorithm. Error metrics used in this case are mean reprojection error and max reprojection error. Reprojection errors can provide quantitive accuracy measurement. Reprojection error means the distance between the estimated point and the corresponding world point.

As can be seen in Table 4.4, AprilCal algorithm works better when calculating triangulation. This means that AprilCal algorithm is better for 3D reconstruction. Using more points showed that being dependent on one point may give wrong results. AprilCal is again more consistent but AprilCal has the maximum inlier as can be seen with the max reprojection error. This inlier means that Pose Selection handles edges better than AprilCal.

Real-world cameras have lens distortion. One of the purposes of camera calibration is to reduce lens distortion. In order to get rid of lens distortion, one needs to calibrate the camera accurately. Lens distortion can give us results that are not usable. Another test metric in this evaluation tool is calibrating the camera under lens distortion. To generate the necessary poses for this test, a constant lens distortion was added to $k_1$ and $k_2$. Added constant is **1** for both distortion coefficients. You can see the generated poses under lens

Table 4.4: Mean and Max reprojection errors over 329 points

| **Calibration Algorithm** | **Mean Reproj. Error** | **Max Reproj. Error** |
|---|---|---|
| AprilCal | 0.000126 | 0.000586 |
| Pose Selection | 0.000168 | 0.000341 |

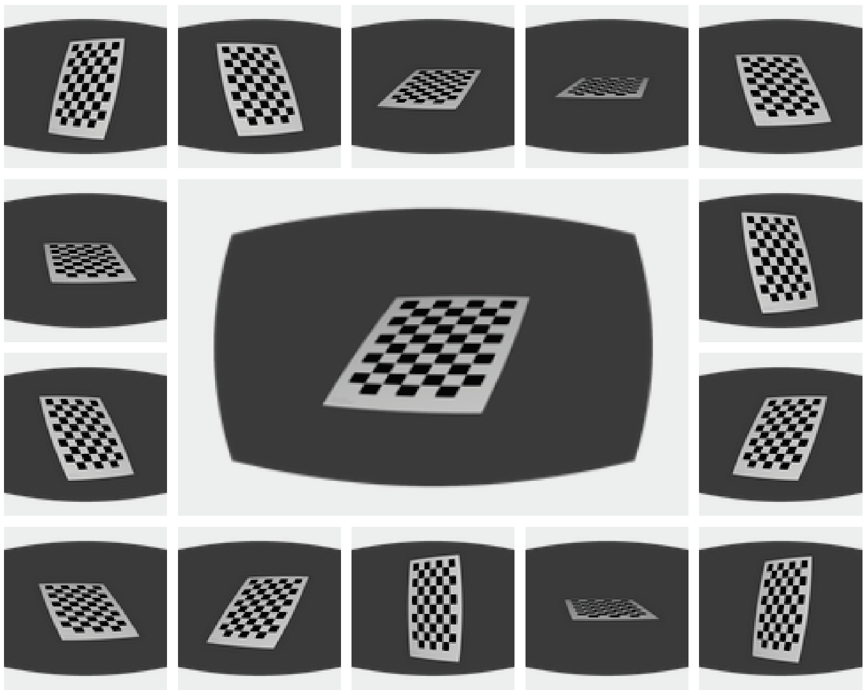


Figure 4.22: Generated AprilCal poses with Blender under distortion

distortion for both calibration algorithms. You can see the generated poses in Figure 4.22 and Figure 4.23

After generating the poses above, the camera calibration process is started. You can see the results for both algorithms Figure 4.24 to Figure 4.29.

For this test case, Mean Reprojection Error results are different. Pose Selection algorithm gives better results. The reason behind this is AprilCal's bootstrap session. In this session, calibration is done with fewer images. Using distorted images affects this step and this can be seen in Figure 4.24 and Figure 4.25. Fewer images under lens distortion gives inaccurate results. However, the camera intrinsics are generated more accurately with AprilCal. In Figure 4.26 to Figure 4.29, the intrinsics for AprilCal are more closer to the ground truth than Pose Selection. This is the result of generated poses and means AprilCal calculates intrinsic parameters more accurately rather than Pose Selection under lens distortion.

To see the acccuracy of the camera calibrations in the real situation, refining the camera matrix and undistorting the picture can help. Refining means controlling the percentage of unwanted pixels in the undistorted image. These steps are done using

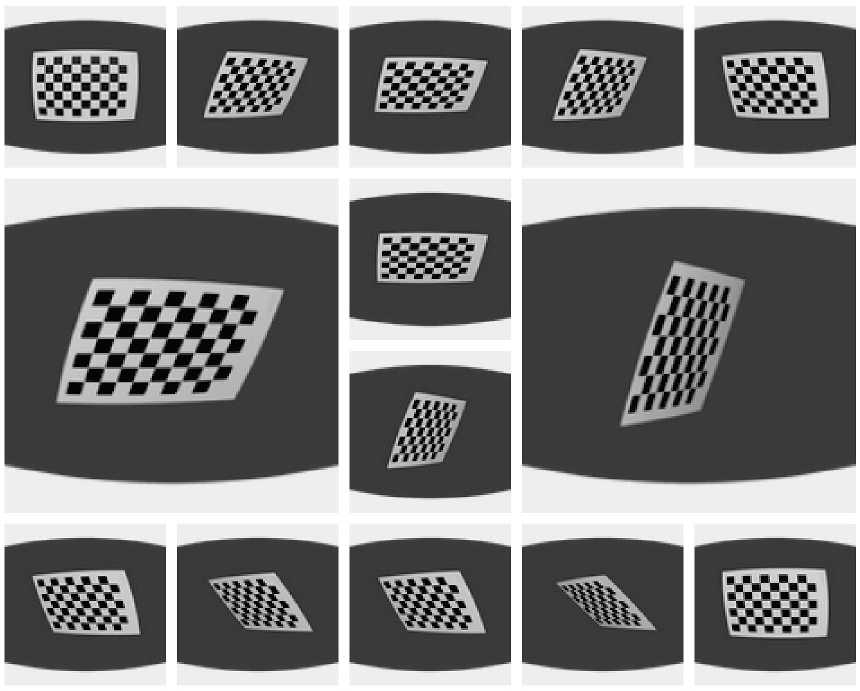


Figure 4.23: Generated PoseCalib poses with Blender under distortion

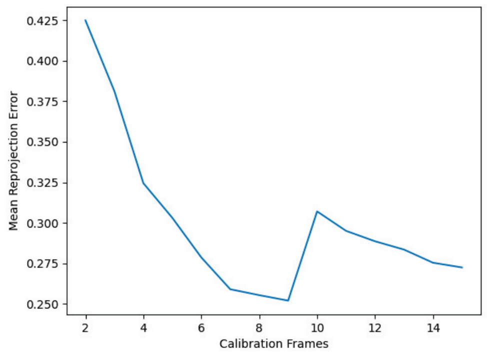


Figure 4.24: AprilCal MRE change under distortion

OpenCV functions. Detailed calculations can be seen in [3]. Undistorted images can be seen in Figure 4.30a and Figure 4.30b.

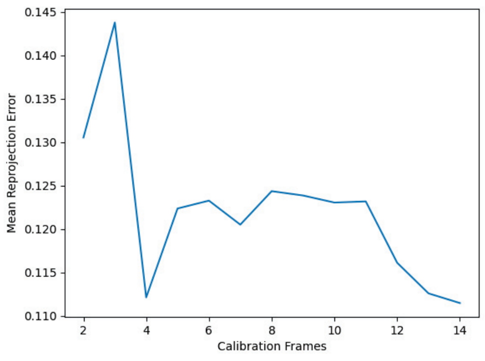


Figure 4.25: Pose Selection MRE change under distortion

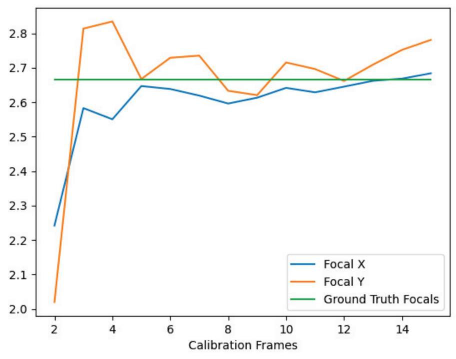


Figure 4.26: AprilCal focal change under distortion

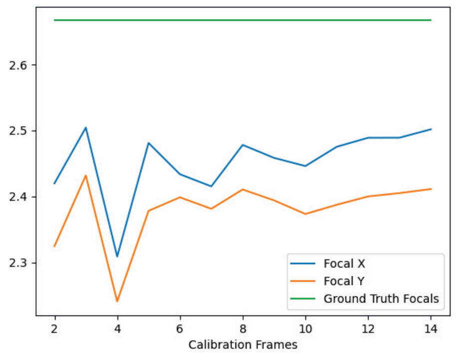


Figure 4.27: Pose Selection focal change under distortion

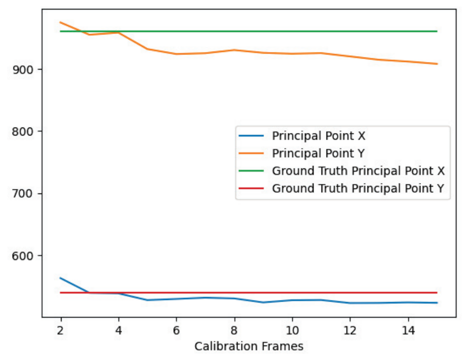


Figure 4.28: AprilCal principal point change under distortion

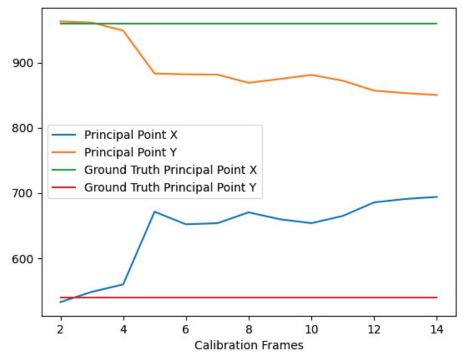
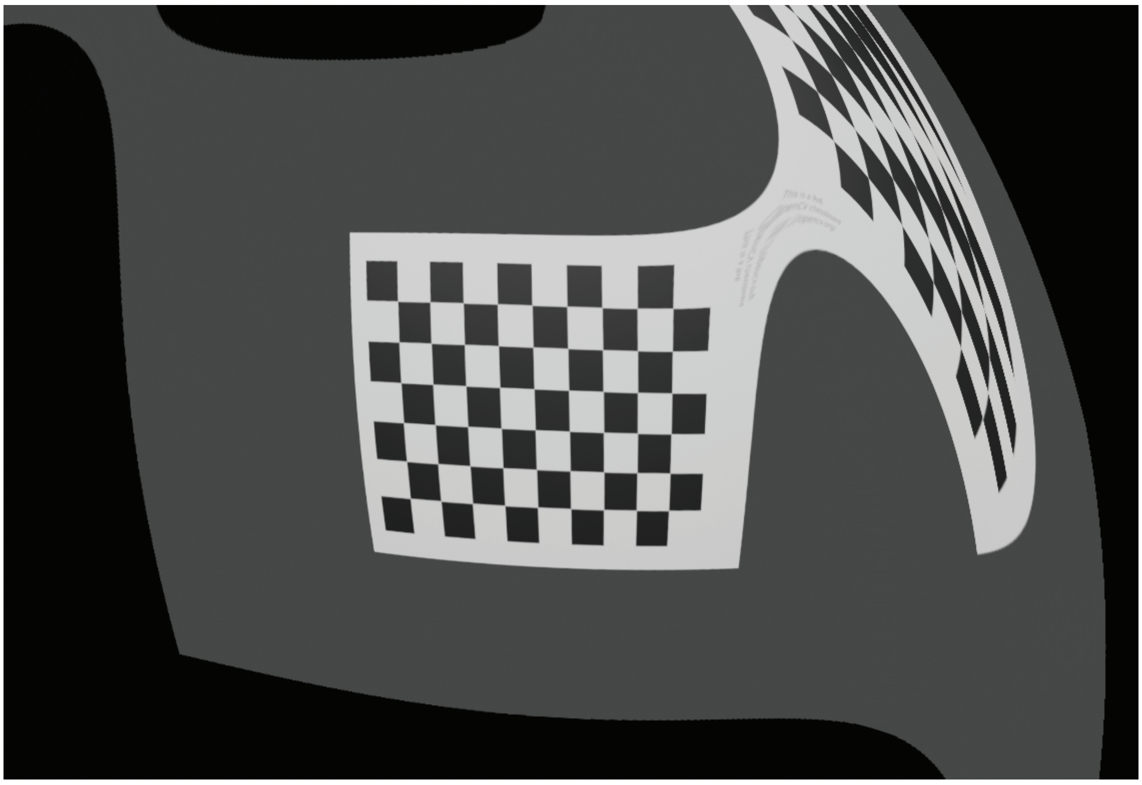


Figure 4.29: Pose Selection principal point change under distortion


(a)

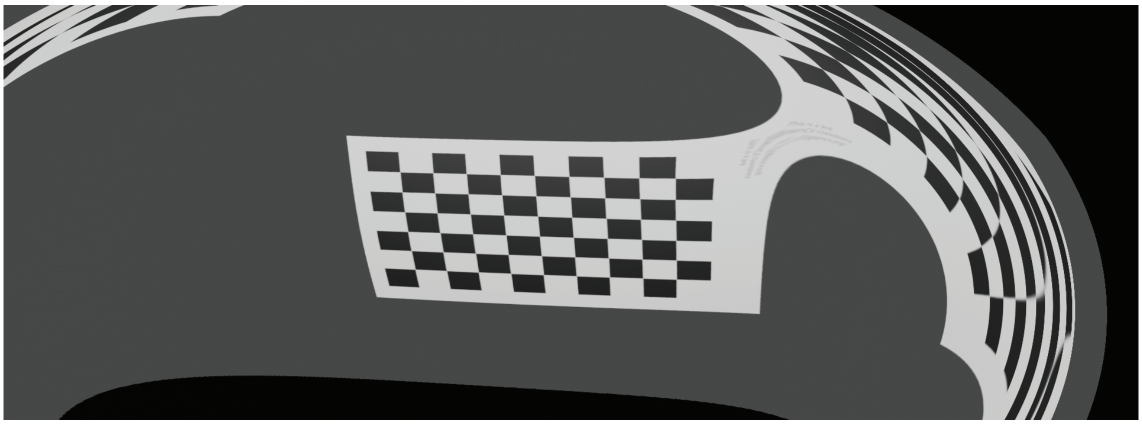

(b)

Figure 4.30: Undistorted images (a) AprilCal (b) Pose Selection

# CHAPTER 5

# CONCLUSION

## 5.1 Summary of the Results

In this study, two different calibration algorithms are explained. An evaluation tool to test these calibration algorithms is proposed. The results of the algorithms are also presented.

In Chapter 3, both calibration algorithms are explained. The results of the algorithms are shared. Their own test results and pose generation steps are presented. For each calibration algorithm, there are different pose generation algorithms presented. One of the calibration algorithms uses specific pattern compositions to avoid degenerate poses. A relation between pose and constraint is described for camera parameters. The covariance of the parameters is used in the calibration. They split the parameters into two groups, pinhole and distortion parameters.

The other calibration algorithm uses new error metrics to generate the new pose. One of their contributions is these error metrics. The estimation of focal length and model uncertainty are calculated with the help of these error metrics. The pose generation steps are also divided into two in this work. In order to create the first poses, the bootstrapping section is used. Then, they try to reduce uncertainties.

In Chapter 4, an evaluation tool to test these calibration algorithms is proposed. This evaluation tool is a 3D modeling tool Blender with Python scripts. With the help of scripts and modeling tool, repeatable tests and outputs are generated. These outputs are used in the comparison of these two well-known calibration algorithms. Each calibration algorithm is tested with this evaluation tool and results are presented. As a result, overall AprilCal is better in consistency and result. With lesser images, AprilCal works better than Pose Selection. AprilCal is more accurate than Pose Selection. In 3D reconstruction, AprilCal is better. Triangulation results are better for AprilCal.

## 5.2 Discussion and Future Work

The proposed evaluation tool creates accurate ground truth data and tests calibration algorithms in a proper way. However, there are numerous improvements to the proposed evaluation tool.

After camera calibrations, the camera can be used in different situations. In football games, autonomous driving cars, and various scenarios. This evaluation tool can be enhanced the test these real-time scenarios. For instance, a 3D-rendered football game scene can be generated to test the camera that will be used in a football game. Real-time obstacles can be generated with the modeling tool to imitate the traffic for autonomous driving car cameras. Having this increases the accuracy and helps us to see the precision of the camera.

Another enhancement is about the pose patterns. Mostly, the checkerboard pattern is used. Including new patterns can create different situations and gives us the ability to test another case. Using different patterns can create different results with calibration algorithms. One can measure the calibration time according to different patterns.

While doing these, there are also some limitations. The proposed evaluation tool uses a 3D modeling tool. The generated poses can only be generated with the modeling tool and we are constrained by it. One can develop a modeling tool for only this purpose.

# REFERENCES

[1] 3d reconstruction using the direct linear transform - bardsley.

[2] Richard hartley's web page.

[3] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

[4] Carlson, A., K. A. Skinner, R. Vasudevan, and M. Johnson-Roberson (2019). Modeling camera effects to improve visual learning from synthetic data. *Lecture Notes in Computer Science*, 505–520.

[5] Chen, J. and J. J. Little (2019). Sports camera calibration via synthetic data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*.

[6] Foundation, B. Home of the blender project - free and open 3d creation software.

[7] Garrido-Jurado, S., R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*(6), 2280–2292.

[8] Hahner, M., D. Dai, C. Sakaridis, J.-N. Zaech, and L. V. Gool (2019). Semantic understanding of foggy scenes with purely synthetic data. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*.

[9] Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant (2020, September). Array programming with NumPy. *Nature 585*(7825), 357–362.

[10] Hartley, R. and A. Zisserman (2019). *Multiple view geometry in Computer Vision*. Cambridge University Press.

[11] Holt, R. J. and A. N. Netravali (1991). Camera calibration problem: Some new results. *CVGIP: Image Understanding 54*(3), 368–383.

[12] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering 9*(3), 90–95.

[13] Lavest, J. M., M. Viala, and M. Dhome (1998). Do we really need an accurate calibration pattern to achieve a reliable camera calibration? *Computer Vision — ECCV'98*, 158–174.

[14] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics 2*(2), 164–168.

[15] Li, S. and B. Lu (2007). Automatic camera calibration technique and its application in virtual advertisement insertion system. *2007 2nd IEEE Conference on Industrial Electronics and Applications*.

[16] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*(2), 91–110.

[17] Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics 11*(2), 431–441.

[18] Martins, P. F., H. Costelha, L. C. Bento, and C. Neves (2020). Monocular camera calibration for autonomous driving — a comparative study. *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*.

[19] MATLAB (2010). *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc.

[20] Olson, E. (2011). Apriltag: A robust and flexible visual fiducial system. *2011 IEEE International Conference on Robotics and Automation*.

[21] Peng, S. and P. Sturm (2019). Calibration wizard: A guidance system for camera calibration based on modelling geometric and corner uncertainty. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

[22] Ren, Y. and F. Hu (2021). Camera calibration with pose guidance. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[23] Richardson, A., J. Strom, and E. Olson (2013). Aprilcal: Assisted and repeatable camera calibration. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[24] Rojtberg, P. and A. Kuijper (2018). Efficient pose selection for interactive camera calibration. *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

[25] Sturm, P. and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*.

[26] Triggs, B. (1998). Autocalibration from planar scenes. *Computer Vision — ECCV'98*, 89–105.

[27] von Neumann-Cosel, K., E. Roth, D. Lehmann, J. Speth, and A. Knoll (2009). Testing of image processing algorithms on synthetic data. *2009 Fourth International Conference on Software Engineering Advances*.

[28] Wang, J. and E. Olson (2016). Apriltag 2: Efficient and robust fiducial detection. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[29] Xu, M., J. Orwell, L. Lowey, and D. Thirde (2005). Architecture and algorithms for tracking football players with multiple cameras. *IEE Proceedings - Vision, Image, and Signal Processing 152*(2), 232.

[30] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*(11), 1330–1334.

# APPENDIX A

## Camera Creator for Blender

```
# Make sure your first camera is named 'Camera'
camera = bpy.data.objects['Camera']
# The camera will face this object. /!\ Naming
target_object = bpy.data.objects['Pattern']

z = camera.location[2]
radius = Vector((camera.location[0],
                        camera.location[1], 0)).length
angle = 2 * math.pi * random.random()

new_camera_pos = Vector((radius * math.cos(angle),
                        radius * math.sin(angle), z))

bpy.ops.object.camera_add(enter_editmode=False,
                        location=new_camera_pos)

# Add a new track to constraint and set it to
# track your object
track_to = bpy.context.object.constraints.new('TRACK_TO')
track_to.target = target_object
track_to.track_axis = 'TRACK_NEGATIVE_Z'
track_to.up_axis = 'UP_Y'

# Set the new camera as active
bpy.context.scene.camera = bpy.context.object
```

# APPENDIX B

## Triangulation

```
first_point = np.array(points.values()[0])
first_rotation, first_jacobian = cv2.Rodrigues(
                                rvecs[list(points.keys())[0]])

second_point = np.array(points.values()[1])
second_rotation, second_jacobian = cv2.Rodrigues(
                                rvecs[list(points.keys())[1]])

RT = np.concatenate([first_rotation,
                                tvecs[list(points.keys())[0]]],
                                axis=-1)
first_projection = np.dot(mtx, RT)

RT = np.concatenate([second_rotation, tvecs[list(
                                points.keys())[1]]], axis=-1)
second_projection = np.dot(mtx, RT)

# TriangulatePoints requires 2xn arrays, so transpose
# the points
p = cv2.triangulatePoints(first_projection,
                                second_projection,
                                first_point.T,
                                second_point.T)

# However, homogeneous point is returned
p = p.astype('float')
p /= p[3]
p = p[:3]
print('Projected point:', p.T)
```

# APPENDIX C

## Calibration Calculation

```
self.total_error = 0
objp = np.zeros((self.height * self.width, 3), np.float32)
objp[:, :2] = np.mgrid[0:self.width,
                        0:self.height].T.reshape(-1, 2)
# 3D points in real world
objpoints = []
# 2D points in image plane
imgpoints = []

for image in self.images:
        image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        ret, corners = cv2.findChessboardCorners(image_gray,
                        (self.width, self.height), None)
        if ret:
                objpoints.append(objp)
                corners2 = cv2.cornerSubPix(image_gray,
                        corners, (11, 11), (-1, -1),
                        self.criteria)
                imgpoints.append(corners2)
        else:
                print("Couldn't find corners..")

if len(self.images) >= self.minimum_image_number:
        self.objpoints = objpoints
        self.imgpoints = imgpoints
        ret, mtx, dist, rv, tv = cv2.calibrateCamera(objpoints,
                        imgpoints,
                        (self.camera_width, self.camera_height),
                        None, None)
```

# APPENDIX D

## Getting Calibration Matrix from Blender

```
def get_calibration_matrix_K_from_blender(camd):
    scene = bpy.context.scene
    f_in_mm = camd.lens
    scale = scene.render.resolution_percentage / 100
    resolution_x_in_px = scale * scene.render.resolution_x
    resolution_y_in_px = scale * scene.render.resolution_y
    sensor_size_in_mm = get_sensor_size(camd.sensor_fit,
                        camd.sensor_width, camd.sensor_height)
    sensor_fit = get_sensor_fit(
        camd.sensor_fit,
        scene.render.pixel_aspect_x * resolution_x_in_px,
        scene.render.pixel_aspect_y * resolution_y_in_px
    )
    pixel_aspect_ratio = scene.render.pixel_aspect_y
                        / scene.render.pixel_aspect_x
    if sensor_fit == 'HORIZONTAL':
        view_fac_in_px = resolution_x_in_px
    else:
        view_fac_in_px = pixel_aspect_ratio * resolution_y_in_px
    pixel_size_mm_per_px = sensor_size_in_mm / f_in_mm
                        / view_fac_in_px
    s_u = 1 / pixel_size_mm_per_px
    s_v = 1 / pixel_size_mm_per_px / pixel_aspect_ratio

    # Parameters of intrinsic calibration matrix K
    u_0 = resolution_x_in_px / 2 - camd.shift_x * view_fac_in_px
    v_0 = resolution_y_in_px / 2 + camd.shift_y * view_fac_in_px
                        / pixel_aspect_ratio
    skew = 0 # only use rectangular pixels


    K = Matrix(
        ((s_u, skew, u_0),
        (   0,  s_v, v_0),
        (   0,    0,   1)))
    return K
```

# APPENDIX E

## Getting Extrinsic Parameters from Blender

```
def get_3x4_RT_matrix_from_blender(cam):
    # bcam stands for blender camera
    R_bcam2cv = Matrix(
        ((1, 0,  0),
        (0, -1, 0),
        (0, 0, -1)))

    # Use matrix_world instead to account for all constraints
    location, rotation = cam.matrix_world.decompose()[0:2]
    R_world2bcam = rotation.to_matrix().transposed()

    # Convert camera location to translation vector
    # used in coordinate changes
    # T_world2bcam = -1*R_world2bcam @ cam.location
    # Use location from matrix_world to account for constraints:
    T_world2bcam = -1*R_world2bcam @ location

    # Build the coordinate transform matrix from world
    # to computer vision camera
    R_world2cv = R_bcam2cv@R_world2bcam
    T_world2cv = R_bcam2cv@T_world2bcam

    # put into 3x4 matrix
    RT = Matrix((
        R_world2cv[0][:] + (T_world2cv[0],),
        R_world2cv[1][:] + (T_world2cv[1],),
        R_world2cv[2][:] + (T_world2cv[2],)
        ))
    return RT
```