

Development and Optimization of a Two-View Model for Anamorphic Projections on Planar Surfaces

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Van Mai Nguyen Thi

Annandale-on-Hudson, New York
May, 2015

Abstract

An anamorphic projection is an image that is intentionally distorted so that the original image can be seen only when looked at from a certain perspective, or using a special device, for example a mirror. Origins of anamorphosis can be traced back to the 16th century art, but beyond the aesthetic values, anamorphosis has found its uses in many practical settings, such as road signs, and keystone correction. Despite the widespread uses, there is a scarcity of detailed explanation of the mathematical principles behind anamorphosis, and there is hardly any computer software generating the anamorphic projections. The goal of this project is to apply computer vision techniques to create a program that automates the process of generating anamorphic projections. We use a projector-camera system to derive homography mappings between the projector and camera image, and propose a method for generating an optimal anamorphic image for multiple viewers using least squares.

Contents

Abstract	1
Dedication	6
Acknowledgments	7
1 Introduction	8
1.1 Background	8
1.2 Previous Work	10
1.3 Motivation	11
1.4 Summary of Results	12
2 Mathematical Concepts	13
2.1 Pinhole Camera Model	13
2.1.1 Mathematical Description	14
2.2 Homogeneous Representation	15
2.3 2D Transformations	18
2.3.1 Transformations	19
2.3.2 Warping	22
3 Algorithmic Methods	24
3.1 Least Squares Minimization Method	24
3.1.1 Linear Model	26
3.1.2 General Model	27
3.2 Projective Transformation	29
3.2.1 Homography Estimation with Linear Least Squares	31
3.2.2 Homography Estimation with Homogeneous Linear Least Squares . .	33

<i>Contents</i>	3
3.3 Feature Detection Methods	35
3.3.1 Harris Corner Detection	35
3.3.2 Good Features To Track	36
4 Results	37
4.1 Basic Planar Model	37
4.1.1 Pattern Detection	40
4.1.2 Homography Estimation	41
4.1.3 Generating the Anamorphic Image	41
4.1.4 Challenges and Limitations	43
4.2 Improved Planar Model	44
4.2.1 Modified Set-up	46
4.2.2 Detecting the Projection Screen	47
4.2.3 Homography Estimation with Chessboard Corner Detection	48
4.2.4 Repositioning and Resizing the Pre-anamorph	49
4.2.5 Challenges and Limitations	51
4.3 Two-view Anamorphosis	52
4.3.1 Two-view Setup	53
4.3.2 Target Camera Images	55
4.3.3 Homography Estimation	57
4.3.4 Generating the Anamorphic Image	62
5 Future Work	64
Bibliography	66

List of Figures

1.1.1 Hans Holbein's <i>The Ambassadors</i> , 1533	9
1.1.2 Undistorted skull from H. Holbein's <i>The Ambassadors</i>	9
1.1.3 Road sign in the shape of an elongated bike. Left: view from the top; right: view from a driver's perspective [7].	10
2.1.1 Graphical representation of the pinhole camera model. [3]	14
2.3.1 Examples of basic 2D transformations. [13]	19
3.2.1 The homographies between the projector image, camera image, and projec- tion surface.	30
4.1.1 Steps involved in finding the homography H_{PC} : (a) project a rectangular pattern (b) Detect the pattern in the camera image	39
4.1.2 Verify that the anamorph is seen correctly by the camera: (a) project the anamorph (b) The image (white rectangle) is seen correctly in the camera image.	39
4.1.3 Screen shots and camera captures of the program	40
4.1.4 Diagram of the scenario in which the points in the camera image are mapped outside of the projector image: (a) camera image (b) projector image	42
4.1.5 (a) Anamorphic image not fitting inside the projector image. (b) When the anamorphic image in (a) is projected, the camera receives an incomplete image.	43
4.1.6 Incorrectly detected projection corners.	44
4.2.1 Screenshots of the program: (a) Detected chessboard corners. (b) Prepared pre-anamorph. (c) Anamorphic image after warping. (d) Anamorphic image seen correctly by the camera.	45
4.2.2 Detected projection screen corners marked with blue dots.	47

LIST OF FIGURES

5

4.2.3 (a) Original chessboard corners in the projector image. (b) Detected chessboard corners in the camera image.	48
4.2.4 Screenshots of the program	50
4.2.5 A scenario in which the program fails to detect the chessboard corners correctly.	51
4.3.1 The new setup: (a) Camera 1 (b) Projector (c) Camera 2 (d) Projection surface.	54
4.3.2 The new setup and notation for the homographies.	55
4.3.3 New chessboard patterns.	57

Dedication

I dedicate this work to my family.

Acknowledgments

First of all, I would like to thank my advisors, Keith and Jim, for their constant advice, support and encouragement throughout this project. I would like to thank Keith for his enthusiasm and support, from brainstorming ideas to finalizing this project. Thank you Jim for your guidance and contribution. My thanks also go to the Mathematics and Computer Science departments for giving me the foundation to work on this project.

I also want to thank my friends, Thinh, Linh, Jin, for supporting me with my project and the good memories that we have had at Bard. I also thank Georgia, Alexzandra and Eva-Marie for being great suitemates and helping me whenever I needed.

I want to thank Thant Ko Ko for his continuous support and always being there for me.

Finally, I would like to thank my parents for their unconditional love and support throughout my life, and my little sister Ola for never failing to make me laugh.

1

Introduction

1.1 Background

An anamorphosis (or anamorphic projection/image) is an image that is intentionally distorted so that the original image can be recovered only when looked at from a certain point of view or using a special device, for example a mirror. Origins of anamorphosis can be traced back to the 16th century when artists, mathematicians and philosophers, fascinated by the idea of perspective, experimented with the notions of illusion, truth and reality [7]. Some of the notable examples of anamorphosis include Jean-Francois Niceron's methods for geometrical construction that generated multiple types of anamorphic transforms which involved both exact and approximate methods [7,8]. One of the most famous anamorphic paintings is Hans Holbein's *The Ambassadors* (Figure 1.1.1), in which the artist embedded an anamorphic image of a skull (Figure 1.1.2) at the bottom of the painting as a *memento mori*, a recurring motif in many artworks in the Renaissance period.



Figure 1.1.1: Hans Holbein's *The Ambassadors*, 1533

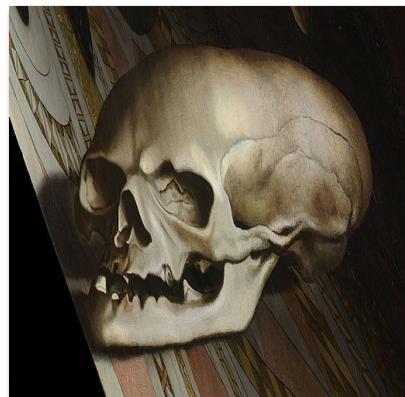


Figure 1.1.2: Undistorted skull from H. Holbein's *The Ambassadors*

Anamorphosis is still prevalent to this day, and is incorporated in many new forms of art, most notably street art. The undying fascination for anamorphosis is manifested in numerous artworks centered around the anamorphic principle and using it to create a unique piece of art. Beyond the aesthetic values, anamorphosis has found its uses in many practical settings, such as road signs that are elongated so that drivers' looking at them from a small angle above the ground can see the signs correctly (Figure 1.1.3), or inverted "ambulance" signs that are designed to be reflected correctly in the rear-view mirrors.

Principles similar to those of anamorphosis are also found in keystone correction, which is a method for eliminating some distortions of projection that may result from misaligning the projector with the projection screen.

Despite the widespread uses, there is a scarcity of detailed explanation of the mathematical principles behind anamorphosis in the context of computer vision, and even with the advanced technology there is hardly any computer software generating the anamorphic projections. The goal of this project is two-fold. First of all, we explore the mathematical principles behind the most common types of anamorphosis. Next, we combine these theories with computer vision techniques to create a program that automates the process of generating anamorphic projections.



Figure 1.1.3: Road sign in the shape of an elongated bike. Left: view from the top; right: view from a driver's perspective [7].

1.2 Previous Work

We build our research on previous works on the topics related to anamorphosis. The mathematical transformations involved in various types of anamorphosis are explained in [7]. In particular, it describes the transformation of points between planar surfaces in perspective anamorphosis, which is the main focus of this project. Because of the lack of resources

about anamorphosis in computer vision, we review other related research projects that employ methods that are applicable to anamorphosis. There have been some development in the area of keystone correction, a topic which greatly overlaps with our project as it uses a similar projector-camera setup and planar image transformation detection methods [2, 12]. Besides keystone correction, further innovations of [2] and [12] include an interactive user experience through a laser pointer. Related but outside of the scope of our project is dynamic anamorphosis, which adapts the generated images to the position of the viewer by tracking the viewer in real-time [9, 11]. We refer to [13] and [6] for comprehensive explanations of fundamental computer vision techniques and methods to analyze images, including feature detection and image transformation. [6] also provides a complete description of the geometric principles behind multiple view systems and techniques for reconstructing a three-dimensional scene using projector camera systems.

1.3 Motivation

While the inspiration for this project originates from the intriguing use of anamorphosis in art, the motivation for our research is the scarcity of previous work exploring the broad applications of anamorphosis in computer vision. Through this project, we hope to contribute to and bring the attention to this overlooked subject.

Most research on the related topics deals with reconstructing the three-dimensional geometry of a scene, for example using a Kinect device or a multiple camera system. This project, however, takes a different approach in that it not only uses the cameras to reconstruct the information about the scene, but also optimizes the projections for the camera views. This work focuses on developing anamorphic models for single and two view geometries, both of which have potential applications in visual and performance arts and can be used to enhance the viewers' experience and create unique and engaging visual effects. We employ anamorphic principles to optimize projected images for a multiple view model,

which has further use in developing anamorphic imaging systems for a large theatrical audience. Another advantage to this simple projector-camera model is its relatively low cost, which means that it can be effectively incorporated into class curricula to facilitate computer vision learning.

1.4 Summary of Results

The results of this project are described in detail in Chapter 4. The findings in this work are built upon a review of the mathematical concepts pertinent to perspective anamorphosis and computer vision techniques, which are employed later in this project.

Chapter 2 discusses the mathematical principles of the Pinhole Camera Model and two-dimensional image transformations, and introduces the concept of homogeneous coordinates. The theoretical background explained in Chapter 2 serves as a basis for computer vision applications and methods described in Chapter 3. Based on what we learn from the review of these concepts, we build three models for generating anamorphic projections on planar surfaces, discussed in Chapter 4. We first implement a basic planar model using a simple projector-camera system. We estimate the homography relations between the images in the system and generate anamorphic images for the camera view. Based on our observations of some limitations in this simple model, we develop an improved model which addresses some of these limitations. These two models are presented in Sections 4.1 and 4.2, and a homography estimation method using linear least squares for single view models is explained in Subsection 3.2.1. Finally, Section 4.3 studies a two view model and Subsection 3.2.2 develops a new method of homography estimation using homogeneous least squares which we implement in Python as part of the algorithm for the two view model.

2

Mathematical Concepts

This chapter defines some basic concepts and terminology needed to describe anamorphic projections. In particular, we present the pinhole camera model and introduce homogeneous representations of geometrical objects used in computer vision to facilitate performing image transformations.

2.1 Pinhole Camera Model

The pinhole camera model is one of the simplest camera models. Its principles were known in as early as the thirteenth century, and used in “camera obscura.” [3]

The pinhole camera can be visualized as a closed box with a small hole on one side, and an image plane on the opposite side. The light rays, which are reflected from objects in the real world, enter the box through the pinhole, and land on the image plane, forming a 180° rotated two-dimensional reflection, as shown in Figure 2.1.1. The same mechanism describes the process of human vision, in which the light passes through the pupil of the eye.

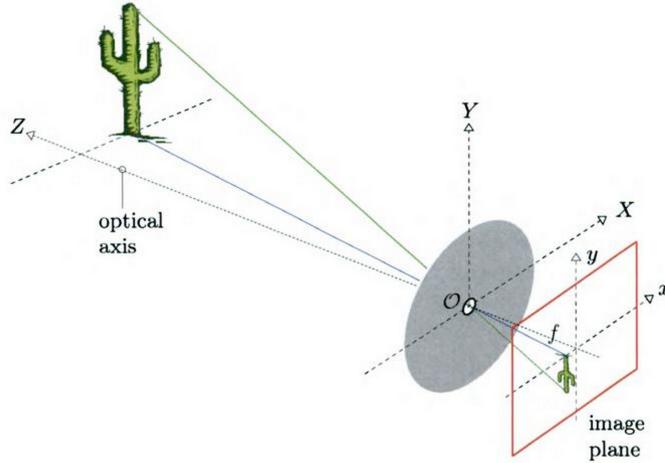


Figure 2.1.1: Graphical representation of the pinhole camera model. [3]

2.1.1 Mathematical Description

For a more rigorous explanation of the model, we introduce a three-dimensional coordinate system where Z is the axis orthogonal to the pinhole and image plane, X is the horizontal axis, and Y is the vertical axis, and the origin is placed at the pinhole, as shown in Figure 2.1.1. The position of any point in the real world in front of the pinhole box can be represented with the X , Y and Z coordinates. The image plane has its own two-dimensional coordinate system with x and y axes parallel to the X and Y axes respectively, and the origin on the Z -axis at a distance f (focal length) from the pinhole.

Using this model, we can easily predict where a point lands on the image plane given its three-dimensional position in the real world. Observe that the ray connecting the point in the real world and its reflection in the image plane forms two similar triangles on either side of the pinhole by projecting itself onto the XZ -plane. Given this observation, we define the position (x, y) of a point on the image plane corresponding to the point (X, Y, Z) in the real world as

$$y = -f \frac{Y}{Z}, \text{ and } x = -f \frac{X}{Z}.$$

2.2 Homogeneous Representation

Homogeneous coordinates were first introduced by August Ferdinand Möbius, and later became a powerful tool with applications in computer vision and projective geometry [1, 10]. The homogeneous system is an alternative to the standard Euclidean, or *inhomogeneous* system as it solves some of its limitations. One limitation of the Euclidean system is that it allows only linear transformations that fix the origin to be represented as matrices. Another limitation is that it does not have a finite representation for a point or line at infinity. The homogeneous model solves this problem by introducing an extra dimension, which provides a convenient representation for points and lines at infinity, as described later in this section. Another benefit of using the homogeneous model, and the reason for its prevalence in computer vision, is that projective transformations can be represented as matrices, which offers a convenient way to perform such operations.

Definition 2.2.1. Let P be a finite point in n dimensions, and (x_1, x_2, \dots, x_n) be its representation in Cartesian coordinates. A *homogeneous representation of point P* is any $(n + 1)$ -tuple $(\omega x_1, \omega x_2, \dots, \omega x_n, \omega)$, where $\omega \neq 0$. The last coordinate ω is called the homogeneous coordinate. A homogeneous representation of a finite point has a non-zero homogeneous coordinate, and a point at infinity has a zero homogeneous coordinate.

The homogeneity of this representation lies in the fact that when a homogeneous point is multiplied by a scalar, it is still the same point and corresponds to the same Cartesian point.

A Cartesian point can be converted to a homogeneous representation by appending $\omega = 1$ as the $(n + 1)^{th}$ coordinate. A homogeneous point can be converted to a Cartesian point by dividing each of its coordinates by ω , and removing the homogeneous coordinate.

Example 2.2.2. Let P_1 be a two dimensional point with Cartesian coordinates (x, y) . A possible homogeneous representation of P_1 is $(x, y, 1)$, or more generally $(x\omega, y\omega, \omega)$ for any $\omega \neq 0$. Let P_2 be another point in two dimensions with homogeneous representation (x, y, ω) . P_2 can be represented in Cartesian coordinates as $(\frac{x}{\omega}, \frac{y}{\omega})$.

Definition 2.2.3. Let l be a straight line in two dimensions defined by the equation $ax + by + c = 0$. Then (a, b, c) is the *homogeneous representation of line l* .

The homogeneity is preserved in this line representation because $(\lambda a, \lambda b, \lambda c)$ represents the line $\lambda ax + \lambda by + \lambda c = 0$, which is the same as line $ax + by + c = 0$ represented by (a, b, c) , for any $\lambda \neq 0$.

Additionally, it follows from the definition of a line that a point P lies on a line l if the dot product of the homogeneous representations of P and l are zero.

Lemma 2.2.4. *A point P is the intersection of two-dimensional lines l_1 and l_2 if and only if P is the cross product of the homogeneous vector representations of l_1 and l_2 .*

Proof. This proof consists of two parts. First, we show that the cross product of the homogeneous representations of two lines is the intersection point of those lines. Then we prove that an intersection of two lines must also be the cross product of the two lines.

Let l_1 and l_2 be two-dimensional non-parallel lines represented as (a_1, b_1, c_1) and (a_2, b_2, c_2) respectively. Their cross product is a point P defined as

$$P = (a_1, b_1, c_1) \times (a_2, b_2, c_2) = (b_1c_2 - b_2c_1, a_2c_1 - a_1c_2, a_1b_2 - a_2b_1). \quad (2.2.1)$$

We verify that P lies on both lines l_1 and l_2 by demonstrating that the dot product of P with each line is zero. For line l_1 we obtain

$$\begin{aligned} & a_1(b_1c_2 - b_2c_1) + b_1(a_2c_1 - a_1c_2) + c_1(a_1b_2 - a_2b_1) \\ &= a_1b_1c_2 - a_1b_2c_1 + a_2b_1c_1 - a_1b_1c_2 + a_1b_2c_1 - a_2b_1c_1 \\ &= 0, \end{aligned}$$

which means that P lies on l_1 . Similarly, we obtain that P lies on l_2 :

$$\begin{aligned} & a_2(b_1c_2 - b_2c_1) + b_2(a_2c_1 - a_1c_2) + c_2(a_1b_2 - a_2b_1) \\ &= a_2b_1c_2 - a_2b_2c_1 + a_2b_2c_1 - a_1b_2c_2 + a_1b_2c_2 - a_2b_1c_2 \\ &= 0. \end{aligned}$$

Since P lies on both l_1 and l_2 , P , which is the cross product of the two lines, is the intersection of those lines.

It remains to show that the intersection of two lines is also the cross product of the lines. Let $Q = (x, y, 1)$ be the intersection point of l_1 and l_2 . Notice that without loss of generality, any homogeneous representation can be transformed to a form with the homogeneous coordinate being 1. Because Q is the intersection point, it must lie on both lines, which means that the dot product of Q with each line is zero. The dot product of Q and l_1 is

$$Q \cdot l_1 = (x, y, 1) \cdot (a_1, b_1, c_1) = a_1x + b_1y + c_1,$$

and the dot product of Q and l_2 is

$$Q \cdot l_2 = (x, y, 1) \cdot (a_2, b_2, c_2) = a_2x + b_2y + c_2.$$

We need to solve the following system of equations for x and y :

$$a_1x + b_1y + c_1 = 0$$

$$a_2x + b_2y + c_2 = 0.$$

We obtain the solution with a simple row reduction:

$$\begin{array}{cc|c} a_1 & b_1 & -c_1 \\ a_2 & b_2 & -c_2 \end{array} \sim \left[\begin{array}{cc|c} 0 & a_2b_1 - a_1b_2 & a_1c_2 - a_2c_1 \\ a_2 & b_2 & -c_2 \end{array} \right]$$

$$\sim \left[\begin{array}{cc|c} 0 & 1 & \frac{a_1 c_2 - a_2 c_1}{a_2 b_1 - a_1 b_2} \\ a_2 & 0 & -c_2 - \frac{a_1 b_2 c_2 - a_2 b_2 c_1}{a_2 b_1 - a_1 b_2} \end{array} \right]$$

$$\sim \left[\begin{array}{cc|c} 0 & 1 & \frac{a_1 c_2 - a_2 c_1}{a_2 b_1 - a_1 b_2} \\ 1 & 0 & \frac{b_2 c_1 - b_1 c_2}{a_2 b_1 - a_1 b_2} \end{array} \right].$$

Therefore the coordinates of Q are

$$\left(\frac{b_2 c_1 - b_1 c_2}{a_2 b_1 - a_1 b_2}, \frac{a_1 c_2 - a_2 c_1}{a_2 b_1 - a_1 b_2}, 1 \right),$$

which in homogeneous representation is equivalent to $(b_2 c_1 - b_1 c_2, a_1 c_2 - a_2 c_1, a_2 b_1 - a_1 b_2)$, which is the cross product of l_1 and l_2 , as computed in the equation (2.2.1). Thus, the intersection point of two lines is also the cross product of the two lines. \square

Definition 2.2.5. The *degrees of freedom* of a system is the number of independent parameters that define the system and are free to vary.

Example 2.2.6. It is clear that a Cartesian point in two dimensions is defined by two parameters: the x and y coordinate. Thus a 2D point has two degrees of freedom. It is less obvious that a line in two dimensions has two degrees of freedom. Recall that a line is defined by an equation of the form $ax + by = c$, where the coefficients a, b and c define the line. Although this equation has three coefficients, a line is unique only up to a scale of the coefficients, that is a line $ax + by = c$ is the same as a line $sax + sbx = sc$ for any non-zero scalar s . In fact, a line can be defined with two independent parameters: the slope and the y -intercept, marked as m and c in this common notation: $y = mx + c$.

2.3 2D Transformations

This section deals with types graphical transformations on 2D images. In computer vision, these operations are functions of pixel coordinates that return new coordinates for the given pixel. The simplest types of transformations include translation, rotation and affine. They form a basis for the projective transformation (homography), which could be thought

of as an advanced transformation that includes but is not limited to a combination of those basic transformations. The projective transformation is defined in this chapter and discussed in more detail in Section 3.2. The procedure of applying a transformation to an image is referred to as warping. There are two main methods of warping, defined in Subsection 2.3.2.

2.3.1 Transformations

This section focuses on some fundamental transformations of two dimensional images, illustrated in Figure 2.3.1. For each transformation, we provide a definition and present a matrix notation that offers a convenient way to apply them to 2D images. Since the transformations discussed here do not preserve the origin, their matrix notation is only possible when we use homogeneous coordinates, introduced in Section 2.2.

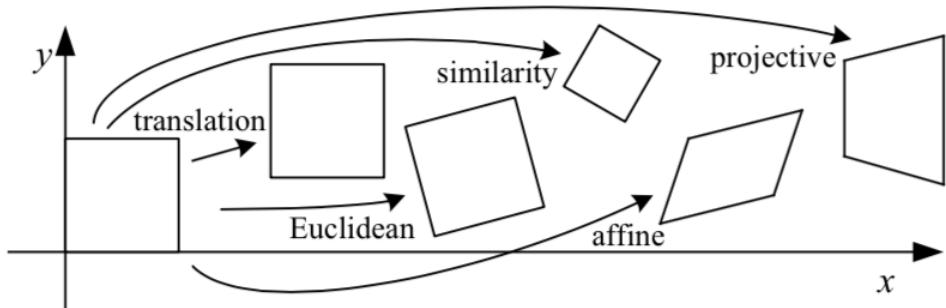


Figure 2.3.1: Examples of basic 2D transformations. [13]

Definition 2.3.1. *Translation* is a type of transformation that shifts an image by a vector.

It is typically defined as a 2×3 matrix

$$T = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix}$$

or as a 3×3 matrix

$$T = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}.$$

for some translation factors t_x and t_y .

The 2×3 matrix can be applied to a homogeneous point this way:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

where $(x, y, 1)$ is initial position, and $(x', y')^T$ is the new position after translation. Observe that the result of translation using the 2×3 matrix is a point in Euclidean coordinates, without the homogeneous coordinate. This creates a significant disadvantage as it prevents combining several transformations. For example, if we want to apply another translation on this result, we need to convert it to the homogeneous coordinates first.

The 3×3 matrix gives the same result, but in homogeneous coordinates:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}.$$

This notation makes it easy to apply another transformation such as translation directly to the result, as the result is already in the homogeneous form.

Definition 2.3.2. *Rotation* is a type of transformation that preserves orientation, lengths and angles. In the most compressed form it is defined as a 2×2 matrix R :

$$R = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

for some angle θ . In this 2×2 form, rotation is applied to an Euclidean point, and the result is also in Euclidean form, and is the position of the initial point rotated by the angle θ around the origin. Another more versatile form of the rotation matrix is

$$R = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

which is applied to a homogeneous point, and returns a rotated homogeneous point.

Definition 2.3.3. *Scaling* is a transformation that preserves orientation and angles. It takes a form of a non-zero scaling factor s , which can be applied to both homogeneous and inhomogeneous points.

Rather than appearing separately, translation, rotation and scaling are often combined to create more complex transformations. For example, a combination of rotation and translation is called *rigid* or *Euclidean* transformation, and is of the form

$$\begin{pmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \end{pmatrix}.$$

A combination of all three basic transformations (translation, rotation and scaling) is referred to as *similarity transform* is defined as

$$\begin{pmatrix} s \cos \theta & -s \sin \theta & st_x \\ s \sin \theta & s \cos \theta & st_y \end{pmatrix}$$

or more generally as

$$\begin{pmatrix} a & -b & t_x \\ b & a & t_y \end{pmatrix}.$$

Definition 2.3.4. An *affine* transformation is a type of transformation that preserves parallel lines. It takes the form of any 2×3 matrix

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{pmatrix}.$$

Definition 2.3.5. A *homography*, also known as *projective transformation* or *perspective transform* is a transformation of a two-dimensional image I to another two-dimensional image I' (referred to as the anamorphic image), such that all straight lines in I are preserved in I' . It is represented as a 3×3 matrix

$$\begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix}.$$

Lemma 2.3.6. A homography H produces the same transformation as a homography λH for any non-zero λ .

Proof. Let $p = (x, y)^T$ be a point in 2D, and H be a homography matrix defined as:

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}.$$

By Definition 2.2.1, $(x, y, 1)^T$ is a homogeneous representation of p . Let point q be the result of the homography transformation H on the homogeneous point p . Then

$$q = Hp = \begin{bmatrix} h_{00}x + h_{01}y + h_{02} \\ h_{10}x + h_{11}y + h_{12} \\ h_{20}x + h_{21}y + h_{22} \end{bmatrix} = \begin{bmatrix} \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \\ \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \\ 1 \end{bmatrix}.$$

The last step of the derivation of q is valid because a homogeneous point remains unchanged when multiplied by a scalar.

Let H' be another homography defined as $H' = \lambda H$ for some non-zero λ . Let q' be the result of the homography H' on p . Then

$$q' = H'p = \begin{bmatrix} \lambda(h_{00}x + h_{01}y + h_{02}) \\ \lambda(h_{10}x + h_{11}y + h_{12}) \\ \lambda(h_{20}x + h_{21}y + h_{22}) \end{bmatrix} = \begin{bmatrix} \frac{\lambda(h_{00}x + h_{01}y + h_{02})}{\lambda(h_{20}x + h_{21}y + h_{22})} \\ \frac{\lambda(h_{10}x + h_{11}y + h_{12})}{\lambda(h_{20}x + h_{21}y + h_{22})} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \\ \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}} \\ 1 \end{bmatrix}.$$

Thanks to the property of scalar invariance of the homogeneous representation, the two transformations H and $H' = \lambda H$ are the same. \square

By Lemma 2.3.6, projective transformation matrices are invariant to scale, and therefore have eight degrees of freedom. This fact is of particular significance as we introduce a method for homography estimation that uses this property in Section 3.2.

2.3.2 Warping

Definitions 2.3.1 through 2.3.5 are different types of two-dimensional transformations. The process of applying a transformation, or a combination of transformations to an image is called warping. We distinguish two basic types of warping: forward and backward, which differ in that forward warping directly transforms the original image to the final image, while backward warping fills in each pixel of the final image with the corresponding color from the original image.

Definition 2.3.7 (Forward Warping). Let t be a transformation function that maps pixels from image I to another image I' . An image can be forward warped by copying the color at each pixel at point $p \in I$ to a point $p' \in I'$, where $p' = t(p)$.

Definition 2.3.8 (Backward Warping). Let t be a transformation function that maps pixels from image I to another image I' , and t^{-1} be the inverse warping that maps pixels from I' to I . An image can be backward warped by iterating through points $p' \in I'$, and “coloring” it with the color at the point $p \in I$, where $p = t^{-1}(p')$.

In practice, warping is often improved by some methods of interpolation that take into account a cluster of pixels to determine the best color for each pixel on the final image, rather than copying a single pixel from the original image. Interpolation is designed to create smoother and more accurate color transitions, and further minimizes the distortions generated by warping. In this project, we use a warping method provided by the OpenCV library which uses bilinear interpolation and nearest-neighbor interpolation [15]. These interpolation methods determine the color value at each pixel by averaging over the pixels around it.

3

Algorithmic Methods

This chapter deals with the techniques and methods used in the implementation of the automated anamorphic projection program that we develop in this project. We introduce the linear least squares minimization method, which is a crucial tool for estimating the projective transformations of various surfaces in our anamorphic model, which we discuss in more detail in Chapter 4. We also explain the main feature detection algorithms such as Harris Corner Detection and Chessboard Corner Detection, which are used to facilitate the process of computing the transformations.

3.1 Least Squares Minimization Method

The least squares method is a technique to estimate the best possible solution to a system of equations, which is especially useful when the exact solution to that system does not exist. This approach is designed to solve overdetermined systems, which means that there are more equations than unknowns. This estimate is obtained by searching for a solution that minimizes the error, defined as the sum of squared differences of the result of the estimated solution and the desired result. One of the most prevalent uses of this method

is in finding the line of best fit to a set of data points, such that the sum of the squared vertical distances from the line to each data point is minimized. The least squares model can be used to predict the outcomes of the points not included in the finite input data. For example, the line of best fit provides a prediction of the outcomes for all points in the domain, and not just the input data points. Because of its application as an error minimization method, the least squares estimation is used in many optimization problems in practical experiments to minimize the errors resulting from inaccuracies of a practical setup. For the same reason, it is of great significance to this project as it is the main technique to estimate projective transformations between the computer screen image, camera image, and the projective screen.

We distinguish two types of least squares methods: linear and non-linear least squares. The linear least squares is an approach for finding the model that best describes a set of input and output data, in which for pair of corresponding input and output data points, the model can be expressed as a linear function of the parameters. Notice that the linear least squares model is not necessarily a linear function in terms of the data points. For example $f(x) = ax^2 + b \log(x)$ is a valid model because $f(x_i) = ax_i^2 + b \log(x_i)$ is linear in terms of the coefficients a and b for all data points x_i . On the contrary, the non-linear model is not linear in terms of the coefficients, although some non-linear models can be transformed into linear models, for example the non-linear model $g(x) = e^{ax}$ can be transformed into $h(x) = \ln(g(x)) = ax$ with the output data being $\ln(y)$ instead of just y . We mentioned that the least squares solves overdetermined systems. This means that for an assumed model with m coefficients a_1, a_2, \dots, a_m we need n input data points, and n corresponding output points, where $n \geq m$.

This section deals with the linear least squares as it is the method used in this project. We demonstrate the method of finding the least squares solution on a simple linear model, and a more general model.

3.1.1 Linear Model

This is an example of the linear least squares on a linear regression model.

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a linear function that models the data with input $\{x_1, x_2, \dots, x_n\}$ and observed output $\{y_1, y_2, \dots, y_n\}$, where $n \geq 2$. The function f is of the form $f(x) = mx + c$ for some coefficients m and c , where $m \neq 0$. In order for f to be a “good” model of the data, we need to pick the values of m and c so that $f(x_i) \approx y_i$ for all $1 \leq i \leq n$.

We define an error function E as $E(m, c) = \sum_{i=1}^n (f(x_i) - y_i)^2$. To shorten the derivations and make it easier to read, we assume that all summations \sum_i in this subsection are equivalent to $\sum_{i=1}^n$. The error function serves as a measure of how well f represents the data. We need to find m and c such that the value of $E(m, c)$ is closest to 0. Because E is always positive (E is a sum of squares), this statement is equivalent to finding m and c at the minimum of E . In order to find the minimum of E , we take the partial derivative of E with respect to each of its arguments, and solve the system of equations in which each derivative is equal to 0. The partial derivative of E with respect to m is

$$\begin{aligned}\frac{\partial}{\partial m} E(m, c) &= \frac{\partial}{\partial m} \sum_i (mx_i + c - y_i)^2 \\ &= \sum_i 2x_i (mx_i + c - y_i) \\ &= \left(\sum_i 2x_i^2 \right) m + \left(\sum_i 2x_i \right) c - \left(\sum_i 2x_i y_i \right).\end{aligned}$$

The partial derivative of E with respect to c is

$$\begin{aligned}\frac{\partial}{\partial c} E(m, c) &= \frac{\partial}{\partial c} \sum_i (mx_i + c - y_i)^2 \\ &= \sum_i 2(mx_i + c - y_i) \\ &= \left(\sum_i 2x_i \right) m + (2n)c - \left(\sum_i 2y_i \right).\end{aligned}$$

We set each derivative to zero and solve the simple system of linear equations in which the first column contains the coefficients of m , and the second column corresponds to the

coefficients of c , and the last column has the constant terms of the resulting equations:

$$\left[\begin{array}{cc|c} \sum_i x_i^2 & \sum_i x_i & \sum_i x_i y_i \\ \sum_i x_i & n & \sum_i y_i \end{array} \right].$$

We perform a simple row reduction to transform it to the reduced row echelon form:

$$\begin{aligned} & \left[\begin{array}{cc|c} \sum_i x_i^2 & \sum_i x_i & \sum_i x_i y_i \\ \sum_i x_i & n & \sum_i y_i \end{array} \right] \\ & \sim \left[\begin{array}{cc|c} n(\sum_i x_i^2) - (\sum_i x_i)^2 & 0 & n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i) \\ \sum_i x_i & n & \sum_i y_i \end{array} \right] \\ & \sim \left[\begin{array}{cc|c} 1 & 0 & \frac{(n \sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{(n \sum_i x_i^2) - (\sum_i x_i)^2} \\ 0 & n & \sum_i y_i - \frac{(n \sum_i x_i y_i)(\sum_i x_i) - (\sum_i x_i)^2(\sum_i y_i)}{(n \sum_i x_i^2) - (\sum_i x_i)^2} \end{array} \right] \\ & \sim \left[\begin{array}{cc|c} 1 & 0 & \frac{(n \sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{(n \sum_i x_i^2) - (\sum_i x_i)^2} \\ 0 & 1 & \frac{\sum_i y_i + \frac{(\sum_i x_i)^2(\sum_i y_i) - (n \sum_i x_i y_i)(\sum_i x_i)}{(n^2 \sum_i x_i^2) - n(\sum_i x_i)^2}}{n} \end{array} \right] \end{aligned}$$

and obtain the optimal values for the coefficients m, c :

$$\begin{aligned} m &= \frac{(n \sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{(n \sum_i x_i^2) - (\sum_i x_i)^2} \\ c &= \frac{\sum_i y_i + \frac{(\sum_i x_i)^2(\sum_i y_i) - (n \sum_i x_i y_i)(\sum_i x_i)}{(n^2 \sum_i x_i^2) - n(\sum_i x_i)^2}}{n} \end{aligned}$$

We substitute these values in the equation $f(x) = mx + c$ to obtain the line of best fit.

3.1.2 General Model

This section presents the linear least squares derivation on a general model.

Let f_1, f_2, \dots, f_m be functions $\mathbb{R} \rightarrow \mathbb{R}$ and let $f : \mathbb{R}^m \rightarrow \mathbb{R}$ be the model function, defined as

$$f(\vec{x}) = a_1 f_1(\vec{x}) + a_2 f_2(\vec{x}) + \cdots + a_m f_m(\vec{x}),$$

where a_1, a_2, \dots, a_m are the coefficients of f . As we mentioned before, f does not need to be a linear function on the variables $\vec{x} \in \mathbb{R}^m$, but f must be linear in terms of its

coefficients. This is important because in the process of finding the least squares solution, the coefficients of f become the unknowns, and the initial arguments of f take values of the known input data, and form the coefficients for the new unknowns. Let the input data be $\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$ and the corresponding observed output be $\{y_1, y_2, \dots, y_n\}$, where $n \geq m$, and $\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{im})^T$ for all $0 \leq i \leq n$. Let X be the matrix of the values that the input data takes in function f , where row i of X contains the values of the input data x_i , defined as:

$$X = \begin{pmatrix} X_{11} & X_{12} & \cdots & X_{1m} \\ X_{21} & X_{22} & \cdots & X_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nm} \end{pmatrix}$$

where $X_{ij} = f_j(\vec{x}_i)$. Notice that X_{ij} is independent of the coefficients a_1, a_2, \dots, a_m , which satisfies the restriction of the linear least squares model.

Let \vec{a} be the vector with all the coefficients of f :

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix},$$

and let \vec{y} be the vector containing the output data:

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Then for each data point x_i , the value of f is $f(x_i) = \vec{X}_i \cdot \vec{a}$, which we want to be equal to y_i , which means that we want $\vec{X}_i \cdot \vec{a} = y_i$. Our system can be represented as $X\vec{a} = \vec{y}$, expanded to:

$$\begin{pmatrix} X_{11} & X_{12} & \cdots & X_{1m} \\ X_{21} & X_{22} & \cdots & X_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nm} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Because this is an overdetermined system, which means that the number of equations is greater than the number of unknowns, the solution to this system might not exist. However,

we can still use the least squares method to estimate a solution for the coefficients that best approximates the input data to the observed output.

We define the error E function as:

$$E(a_1, a_2, \dots, a_m) = \sum_{i=1}^n |f(\vec{x}_i) - y_i|^2 = \sum_{i=1}^n |\vec{X}_i \cdot \vec{a} - y_i|^2 = \|X\vec{a} - \vec{y}\|^2$$

To find the values of a_1, a_2, \dots, a_m that minimize the error E , we find the partial derivatives of E with respect to the parameters a_k and set them equal to zero. The system of linear equations resulting from that operation is equivalent to $X^T X \vec{a} = X^T \vec{y}$. It remains to solve the system for the unknown \vec{a} . The product of the matrices X^T and X is a square matrix with size m , and $X^T \vec{y}$ is a vector of length m , so the number of equations is the same as the number of unknowns, which means that we have an exact solution to the system. The solution a_1, a_2, \dots, a_m defines the model f that best fits the data.

3.2 Projective Transformation

The relation between the anamorphic (distorted) image on a plane and the correct image that the viewer observes is equivalent to that of an image projectively warped and the original undistorted image, and so is simply a homography. This relation has in fact been discussed in some previous research dealing with keystone correction [12]. Although seemingly a separate problem, keystone correction involves the same plane-to-plane homographies, and the results of it can very well be adapted to our anamorphic problem. The setup for this project consists of three surfaces detailed in Table 3.2.1.

Consider the transformation of an image from the projector screen to the projection surface. Because the projection surface is a plane, this transformation is a homography. Similarly, the transformation on the image from the projection surface to the camera screen is also a homography because both of these surfaces are planar. It follows that the

Surface	Explanation
camera image/screen	image seen by the camera, also interpreted as the image seen by the viewer
projector image/screen	image projected by the projector; this is the image that the computer gives to the projector, and not the projected image on the projection surface
projection surface	the actual projection on the screen (e.g. wall), with real-life coordinates as opposed to the virtual coordinates of the camera and projector image

Table 3.2.1: An explanation of the naming convention of the surfaces used in this project.

transformation from the projector screen to the camera screen is simply a composition of the two homographies, and therefore is also a homography.

We differentiate between different homographies by introducing the following notation:

H_{AB} is a homography mapping from surface A to surface B , where $A, B \in \{C, P, S\}$ with C, P, S corresponding to camera, projector, and projection surface respectively, represented on the diagram in Figure 3.2.1. For example, H_{PC} is the homography mapping each point in the projector image to a point on the camera image.

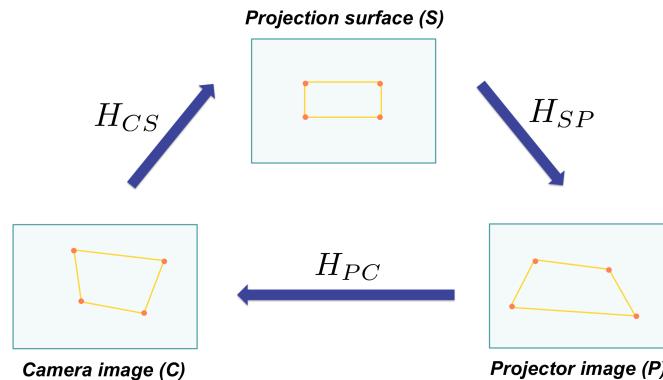


Figure 3.2.1: The homographies between the projector image, camera image, and projection surface.

3.2.1 Homography Estimation with Linear Least Squares

This subsection explains a method of homography estimation for a simple anamorphic model using the least squares. While there are various methods of homography estimation, for example the approach suggested in the *Smarter Presentations* model by Sukthankar et al. [12], we present this method of estimating the homography as an illustrative example.

To simplify the notation, let $H = H_{PC}$ be the mapping from the projector to camera plane, as shown in Figure 3.2.1. Its inverse, H^{-1} , maps from the camera image back to the projector image. The experimental design excludes the possibility of the last entry of the matrix H being zero. This is because a homography with zero as the last entry maps the point at the origin to a point at infinity, as demonstrated in this example:

$$\begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{02} \\ h_{12} \\ 0 \end{pmatrix}.$$

This case is not applicable to this project because our controlled experiment ensures that all relevant points, including the origin, are mapped to points that are finite in Euclidean coordinates.

We formulate the problem as follows. Let $\{p_0, p_1, \dots, p_{n-1}\}$ be a set of points in the projector image, for some $n \geq 4$. The reason for n to be at least 4 is explained later in this section. Each point p_i can be expressed in homogeneous coordinates as $p_i = (x_i, y_i, 1)^T$. Let $\{q_0, q_1, \dots, q_{n-1}\}$ be the set of points in the camera image such that q_i corresponds to p_i for all $0 \leq i \leq n - 1$. Each point q_i can be expressed in homogeneous coordinates as $q_i = (X_i, Y_i, 1)^T$. Since the points in the projector image are transformed to points in the camera image by projective transformation, any point p_i is transformed into Hp_i in the camera image, where H is the homography mapping the projector to the camera image. Without loss of generality, we assume that the last entry of H is 1, so H is of the form:

$$H = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{pmatrix}$$

for some unknown entries $h_{00}, h_{01}, \dots, h_{21}$. It follows that for any point p_i , we have

$$H p_i = \begin{pmatrix} h_{00}x_i + h_{01}y_i + h_{02} \\ h_{10}x_i + h_{11}y_i + h_{12} \\ h_{20}x_i + h_{21}y_i + 1 \end{pmatrix} = \begin{pmatrix} \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + 1} \\ \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + 1} \\ 1 \end{pmatrix},$$

where the last step is true by the Definition 2.2.1 of the homogeneous point representation.

Our goal is to find the values of $h_{00}, h_{01}, \dots, h_{21}$ such that $H p_i \approx q_i$ for all $0 \leq i \leq n - 1$.

This can be expressed as:

$$\begin{pmatrix} \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + 1} \\ \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + 1} \\ 1 \end{pmatrix} = \begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix}. \quad (3.2.1)$$

We transform this system of non-linear equations into an equivalent system of linear equations:

$$\begin{pmatrix} h_{00}x_i + h_{01}y_i + h_{02} \\ h_{10}x_i + h_{11}y_i + h_{12} \\ 1 \end{pmatrix} = \begin{pmatrix} X_i(h_{20}x_i + h_{21}y_i + 1) \\ Y_i(h_{20}x_i + h_{21}y_i + 1) \\ 1 \end{pmatrix}. \quad (3.2.2)$$

For each point p_i and the corresponding point q_i , we have two constraints:

$$h_{00}x_i + h_{01}y_i + h_{02} = X_i(h_{20}x_i + h_{21}y_i + 1)$$

$$h_{10}x_i + h_{11}y_i + h_{12} = Y_i(h_{20}x_i + h_{21}y_i + 1),$$

which can be rewritten in terms of all variables $h_{00}, h_{01}, \dots, h_{21}$:

$$x_i h_{00} + y_i h_{01} + h_{02} + 0h_{10} + 0h_{11} + 0h_{12} - X_i x_i h_{20} - X_i y_i h_{21} = X_i$$

$$0h_{00} + 0h_{01} + 0h_{02} + x_i h_{10} + y_i h_{11} + h_{12} - Y_i x_i h_{20} - Y_i y_i h_{21} = Y_i.$$

Combining all equations resulting from all n point correspondences, we obtain the following system of linear equations:

$$\begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -X_0 x_0 & -X_0 y_0 \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -Y_0 x_0 & -Y_0 y_0 \\ \vdots & \vdots \\ x_{n-1} & y_{n-1} & 1 & 0 & 0 & 0 & -X_{n-1} x_{n-1} & -X_{n-1} y_{n-1} \\ 0 & 0 & 0 & x_{n-1} & y_{n-1} & 1 & -Y_{n-1} x_{n-1} & -Y_{n-1} y_{n-1} \end{pmatrix} \begin{pmatrix} h_{00} \\ h_{01} \\ \vdots \\ h_{21} \end{pmatrix} = \begin{pmatrix} X_0 \\ Y_0 \\ \vdots \\ X_{n-1} \\ Y_{n-1} \end{pmatrix}.$$

Observe that n point correspondences give us $2n$ equations, and we have eight unknowns h_{ij} , so we need $n \geq 4$ so that the number of equations is at least the number of unknowns.

When $n > 4$, our system is overdetermined, which means that there is no exact solution to it; in that case we use the least squares method. To shorten the notation of the system, let A denote the matrix of coefficients, \vec{h} denote the vector of unknowns, and \vec{b} denote the vector of X_i and Y_i coordinates of points q_i , so that our system of equations is $A\vec{h} = \vec{b}$.

We define the error function E as:

$$E(h_{00}, h_{01}, \dots, h_{21}) = \sum_{i=0}^{2n-1} |A_i \cdot \vec{h} - b_i|^2 = \|A\vec{h} - \vec{b}\|^2.$$

We need to find \vec{h} that minimizes E , or in other words, \vec{h} at the minimum of function E . To compute the minimum of E , we solve a system of equations in which the partial derivatives of E are equal to zero. This is equivalent to solving $A^T A \vec{h} = A^T \vec{b}$. Finally, we populate the entries $h_{00}, h_{01}, \dots, h_{21}$ of the homography H with the entries of \vec{h} , and the last entry of H with 1.

3.2.2 Homography Estimation with Homogeneous Linear Least Squares

The homography estimation method presented in the previous Subsection 3.2.1 is a convenient way to transform the nonlinear homography estimation problem into a linear one. Recall that the goal of the previous estimation was to find the entries $h_{00}, h_{01}, \dots, h_{21}$ of the homography H that give the best solution to the system of nonlinear equations (3.2.1) for all point correspondences. Finding the best solution to that system is equivalent to minimizing the error function E defined as:

$$E = \sum_i \left(\frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + 1} - X_i \right)^2 + \left(\frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + 1} - Y_i \right)^2.$$

Although this system is nonlinear, it can be easily transformed into an equivalent linear system with the error function:

$$\begin{aligned} E = \sum_i & (h_{00}x_i + h_{01}y_i + h_{02} - X_i(h_{20}x_i + h_{21}y_i + 1))^2 \\ & + (h_{10}x_i + h_{11}y_i + h_{12} - Y_i(h_{20}x_i + h_{21}y_i + 1))^2. \end{aligned}$$

This technique was used in the previous homography estimation method to create the linear system, as shown in equation 3.2.2.

While this method offers a simplification to the nonlinear homography estimation problem and allows us to solve the problem with the linear least squares, it is not the only way to approach this problem. This section introduces another method for estimating the homography using the homogeneous linear least squares. We begin with formulating the goal of this problem: we need to find a homography H such that for every point p_i on the projector image and the corresponding point q_i in the camera image, Hp_i is approximately equal to q_i . An intuitive yet incorrect representation of the goal statement is $Hp_i \approx q_i$. It is false because it fails to take into account the fact that in homogeneous representation, q_i represents the same point as λq_i for any non-zero λ , so the best solution H might approximate Hp_i to some multiple of q_i . To address this problem we introduce a modified equation $Hp_i \approx \lambda_i q_i$, where λ_i is an auxiliary scaling variable for every point q_i . To find the best solution to this problem, we calculate H and λ_i for $0 \leq \lambda \leq n - 1$ that minimize the error function

$$E = \sum_i \|Hp_i - \lambda_i q_i\|^2.$$

By the linear least squares method, we set the partial derivatives of E to zero, and solve the resulting system of linear equations to find the best H and λ_i , and then use the H to compute the anamorph.

3.3 Feature Detection Methods

Feature detection is one of the most crucial tools to this project. Since our goal is to create a system that automatically generates anamorphic images, we need a way to tell the program about its surroundings, and more specifically about the surface that it needs to project the anamorphic image on. A convenient way to gather information about the scene is by using a camera to capture the necessary details of the scene. However, while a person is able to easily recognize and differentiate between objects in the scene as soon as he/she sees it, the image of the scene is nothing more than a collection of pixels to a computer program. To help the computer understand visual signatures of the scene, we tell it to find some unique characteristics in the image based on the distribution and intensity of the pixels. For example, analyzing an image in terms of the difference in intensity of the neighboring pixels gives us information about the location of potential edges in the image. In the context of this project, this is one way to detect the position of the projection screen. Besides edges, other common easily detectable features include corners and blobs, and these will prove especially useful in this project. In this section, we explore some methods of feature and pattern detection, including the popular Harris Corner Detection, and its improved version called Good Features to Track.

3.3.1 *Harris Corner Detection*

The Harris Corner Detection is one of the most common methods of feature detection. It takes advantage of the observation that the change in intensity of the pixels around a corner, relative to that corner is relatively large, so corners can be easily detected by searching for pixels with the greatest variance in intensity of the pixels around it. This procedure can be visualized as shifting a small window across an image, stopping at every pixel and counting the sum of squared differences of the intensity of that pixel and each pixel that is around it and inside the window.

We present the method based on the publication of Harris and Stephens in 1988 [5]. Let I be an image. Without loss of generality, we assume that the image is in gray scale. Let $I_{x,y}$ denote the intensity, or gray scale color, of the pixel at position (x,y) on the image. Let E be the change in intensity resulting from a shift (u,v) , defined as:

$$E_{u,v} = \sum_{x,y} w_{x,y} |I_{x+u,y+v} - I_{x,y}|^2$$

where $w_{x,y}$ is the window with unity in the desired region, and zero elsewhere.

A corner occurs when the change E is large in two dimensions, and is classified as a relevant feature if the value of E at that corner is above some threshold value.

3.3.2 Good Features To Track

The Good Features to Track algorithm was developed by Shi and Tomasi [14] as an improvement to the Harris corner detection, and a tool to facilitate feature-based vision systems that require excellent tracking of features from frame to frame. This algorithm is an improvement from the earlier feature detection methods in that it utilizes new tracking algorithms [14, 15].

4

Results

This chapter describes the implementation and results of different models of perspective planar anamorphosis. Specifically, the chapter presents the experimental setup, procedures, and limitations for each model as well as explains the algorithms and main functions utilized in the implementations. This chapter begins with a simple planar model that serves as a platform for implementing more complex types of anamorphosis. Due to the iterative nature of the project, later sections integrate functions and computations derived in preceding sections as appropriate. Unless otherwise indicated, these functions serve the same purpose as they first appear in the chapter.

4.1 Basic Planar Model

The goal of this section is to build a program that generates an anamorphic image on a planar surface. The only equipment needed is a projector and camera connected to a computer, and a blank flat surface. The setup for this model consists of three surfaces: camera image, projector image, projection surface, defined earlier in Table 3.2.1. For simplicity purposes, we make several assumptions before the experiment. First, we assume that the

camera is the viewer in order to avoid the necessity to define the position of the viewer, and to make it easier to record the image seen by the viewer. Second, we assume that the camera and projector are both pointed at the projection surface (although preferably from different locations) so that the projector image is within the projection surface and the camera's scope encompasses the entire projection surface. It is also advised to keep dim lighting because it makes the projection more distinguishable from the background of the projection surface.

In order to maintain the consistency of the terms used in this chapter, we also introduce the following naming convention:

Name	Explanation
pre-anamorph	the original image that will be used for warping
anamorph or anamorphic image	the warped image that looks askew before projecting, but appears "straight" when viewed by the viewer (or the camera)

Table 4.1.1: An explanation of the naming convention for the anamorphic and pre-anamorphic image.

The main task in generating the anamorphic image is finding the relevant homographies that, when applied to the pre-anamorph, produce an anamorph that appears correctly in the camera image. The algorithm for generating the anamorphic image is outlined in the following four steps:

1. Project a pattern.
2. Find at least four common points in the projector and camera image.
3. Estimate homography H_{PC} using the least squares method.
4. Use $H_{CP} = H_{PC}^{-1}$ to warp the original image.

Figure 4.1.1 provides a conceptual diagram illustrating this algorithm. In step 1. of the algorithm, we project a simple rectangular pattern (Figure 4.1.1a), and capture the

projection with the camera (Figure 4.1.1b). We find the homography H_{PC} using the point correspondences in the projector and camera image, and use H_{PC} to create the anamorphic image. Figure 4.1.2 illustrates the steps involved in projecting the anamorphic image and verifying that it is seen correctly by the camera.

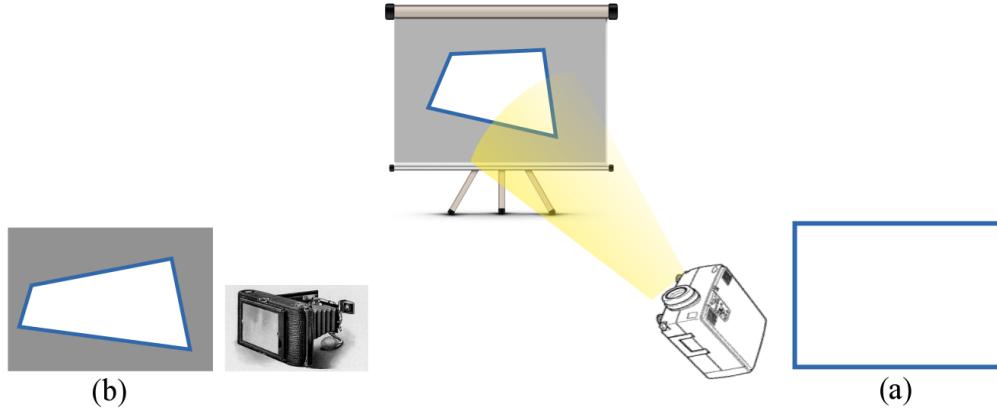


Figure 4.1.1: Steps involved in finding the homography H_{PC} : (a) project a rectangular pattern (b) Detect the pattern in the camera image

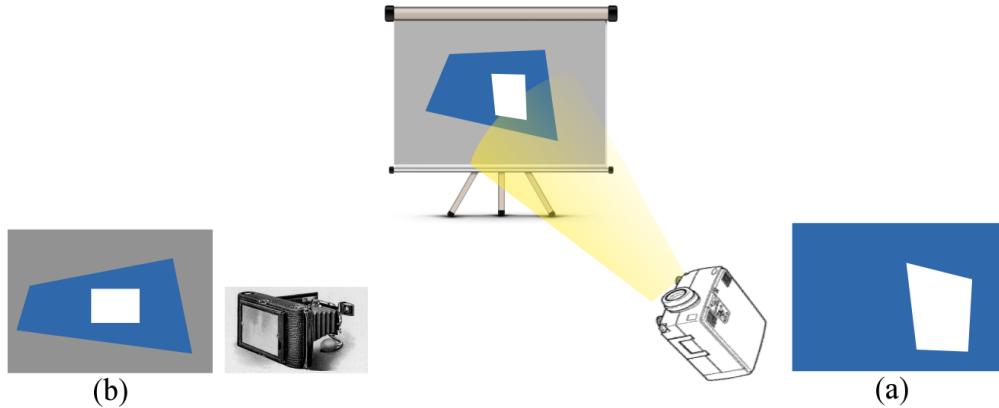


Figure 4.1.2: Verify that the anamorph is seen correctly by the camera: (a) project the anamorph (b) The image (white rectangle) is seen correctly in the camera image.

The result of the algorithm for generating the anamorphic image is shown in the screen shots and camera captures in Figure 4.1.3. The camera captures of the projection are taken with the long exposure technique, which takes multiple pictures from the same position and averages over them to reduce the blinking effect of the projector. A detailed explanation

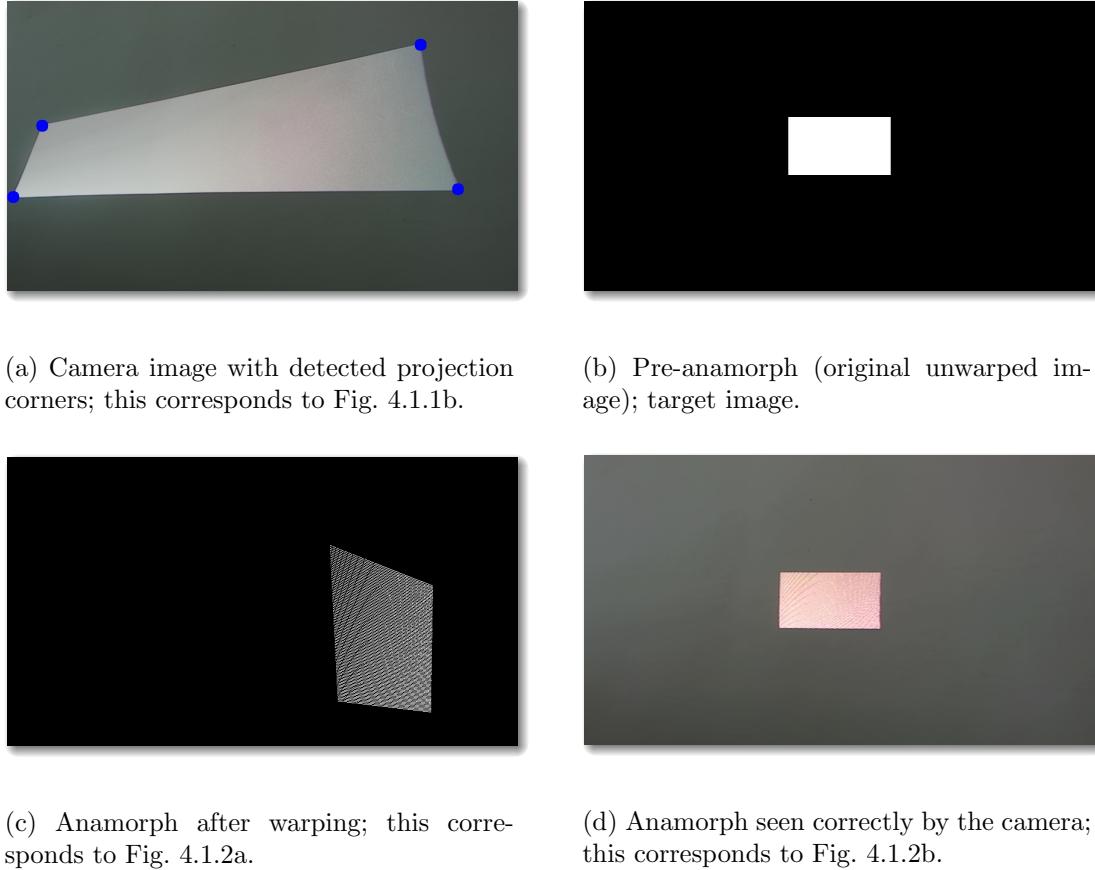


Figure 4.1.3: Screen shots and camera captures of the program

of each step of the algorithm, and a discussion of the challenges and limitations of this model are provided in the following subsections.

4.1.1 Pattern Detection

A simple white rectangle is projected on the projection surface, and captured by the camera. Next, the corners of the projected rectangle are detected in the camera image using the Good Features to Track algorithm, described in Section 3.3.2. This algorithm is convenient in that it allows us to specify the number of corners to be detected, and returns that number of the “best” corners. The four corners of the rectangular projection are correctly detected and marked with blue circles in the camera image as shown in 4.1.3a.

4.1.2 Homography Estimation

The corners of the projection detected in the previous subsection (4.1.1) are arranged in the clockwise order starting from the top-left corner. The ordered corners from the camera image are then paired up with the corners of initial pattern in the projector image, which in this case are simply the outermost corners of the projector image, arranged in the same clockwise order. We use the homography estimation method available in the OpenCV Python library to obtain the best homography H_{PC} relating the projector image to the camera image. The OpenCV estimation function gives the same result as the homography estimation method described in Section 3.2.1.

4.1.3 Generating the Anamorphic Image

The ultimate goal of this model is to obtain the undistorted original image in the camera. The question is: What should we project to obtain correct the projective distortion of the camera? The answer to this question becomes clear if we reverse our perspective. So far, we have considered the transformation of images from the projector image to the camera image. However, in this problem, the camera image is the given factor (we know that the camera should obtain the undistorted original image), and the projector image is the unknown that we need to solve for.

It follows that the anamorphic image (which will be projected) can be generated by applying forward warping on the original image (Figure 4.1.3b), as it is exactly the target image that we want to receive in the camera. For every pixel in the original image, we apply H_{CP} to its homogeneous position $(x, y, 1)$ to obtain a new position (x', y', ω) in the projector image, and copy the pixel to that new location. The result is an anamorphic image that looks distorted in the projector (computer) screen (Figure 4.1.3c), but appears correct once projected and captured in the camera image (Figure 4.1.3d).

One technical issue that we encounter in this step of the algorithm as a result of the setup of this experiment is that some parts of the camera image correspond to the area outside of the projector image. For example, if we apply the homography H_{CP} directly to the original image (pre-anamorph), then some parts of the resulting anamorphic image will be outside of the projector image. Instead we must choose a target area within the camera view that lies inside the projection area.

Figure 4.1.4 illustrates a situation in which a point in the camera image is transformed to a point beyond the projector screen. The gray rectangle in the left (a) is the camera image, and the white quadrilateral shape inside the rectangle is the projection area. This quadrilateral projection area in the camera image corresponds to the projector image, represented as the white rectangle on the right (b). A point p lies outside of the projection area in the camera image (a). When the entire camera image is warped to the projector image, the point p is transformed to a point p' in the projector image. Because p lies outside of the projection area, the point p' must lie outside of the projector image.

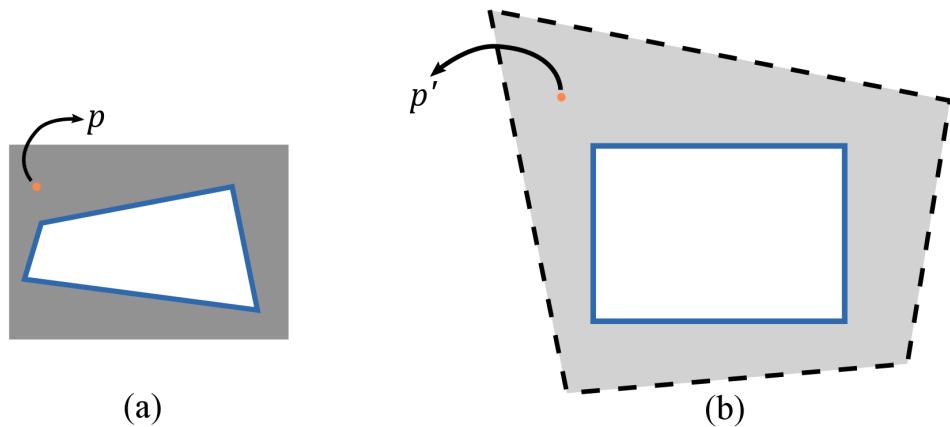


Figure 4.1.4: Diagram of the scenario in which the points in the camera image are mapped outside of the projector image: (a) camera image (b) projector image

A temporary solution to this problem is to reduce the size of the original image by some fixed factor, and place it in the center of the potential (target) camera image, which

is some estimate/guess of where the projection lands in the camera image. If the scale factor is small enough, this solution would work for most projector-camera arrangements. However, this method does not guarantee that the anamorph fits inside the projector image for all possible projector and camera positions, and might significantly decrease the size of the resulting anamorphic image. We will describe a better solution to this problem in the Improved Model discussed in the next section (4.2).

4.1.4 Challenges and Limitations

As explained in Subsection 4.1.3, the current algorithm for generating the anamorphic image might produce an image that does not fit inside the projector image. An example of this is shown in Figure 4.1.5.

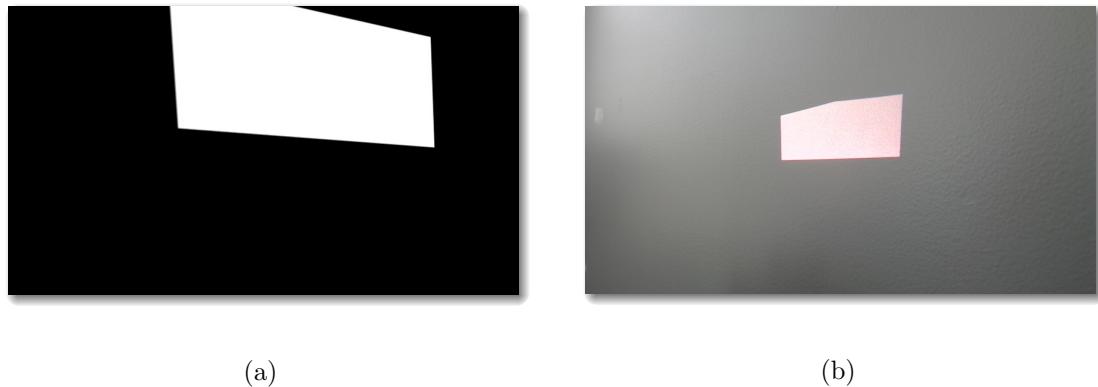


Figure 4.1.5: (a) Anamorphic image not fitting inside the projector image. (b) When the anamorphic image in (a) is projected, the camera receives an incomplete image.

Figure 4.1.6 shows an example of incorrectly detected projection corners, which prevented the program from generating a correct anamorphic image. This error might be caused by poor lighting of the system, which affects the contrast between the projection and the projection surface, making it less detectable by the corner detection function. In some situations, the projection corners are not detected due to the position of the camera, which might be at an angle relative to the projection such that some parts of the projection do not appear clearly in the camera image.

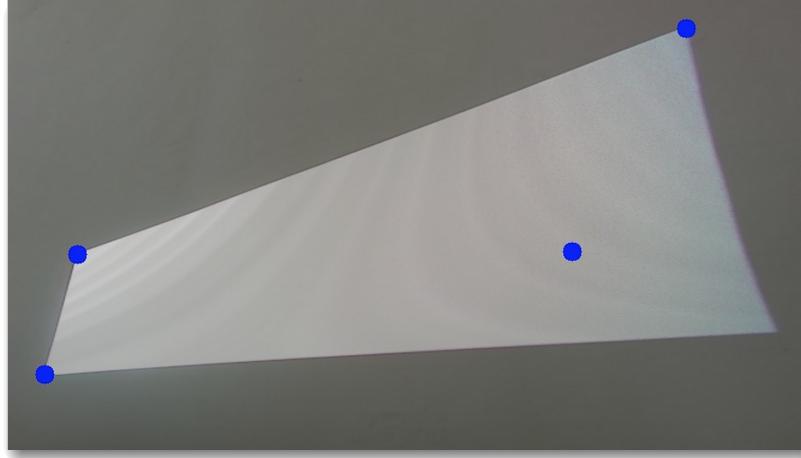


Figure 4.1.6: Incorrectly detected projection corners.

4.2 Improved Planar Model

The Basic Model described in the previous section (4.1) presents a simple method for generating an anamorphic image based solely on the homography between the projector and camera image. While effective and sufficient in the simple scenario of our project, the said method has three major limitations. First, it neglects the relationship between the physical projection surface to the virtual projector and camera images. Although this missing relationship does not affect the process of generating the anamorphic images when using a camera as the viewer, it is necessary for potential future directions of this project. In particular, the homographies mapping to the projection surface are crucial if we want to generate an anamorphosis for a real person as the viewer instead of the camera, and allow us to easily adapt this system to create a keystone correction model. Second, the previous method also fails to reliably eliminate the case of the anamorphic image not fitting inside the projector image. Lastly, the pattern detection method described in Subsection 4.1.1 is deficient because it supplies only four point correspondences, which is the minimum number of correspondences required to estimate the homography H_{PC} . When the conditions such as lighting, and projector and camera angle are unfavorable, this limitation hinders

the process of homography estimation because the error from detecting each of the four corners of the rectangular pattern projected has a significant effect on the accuracy of the resulting homography. This section uses a modified set-up to incorporate the projection surface, proposes a more complex pattern detection algorithm, and introduces a method for repositioning of the pre-anamorph to guarantee that it fits inside the projector image after warping.

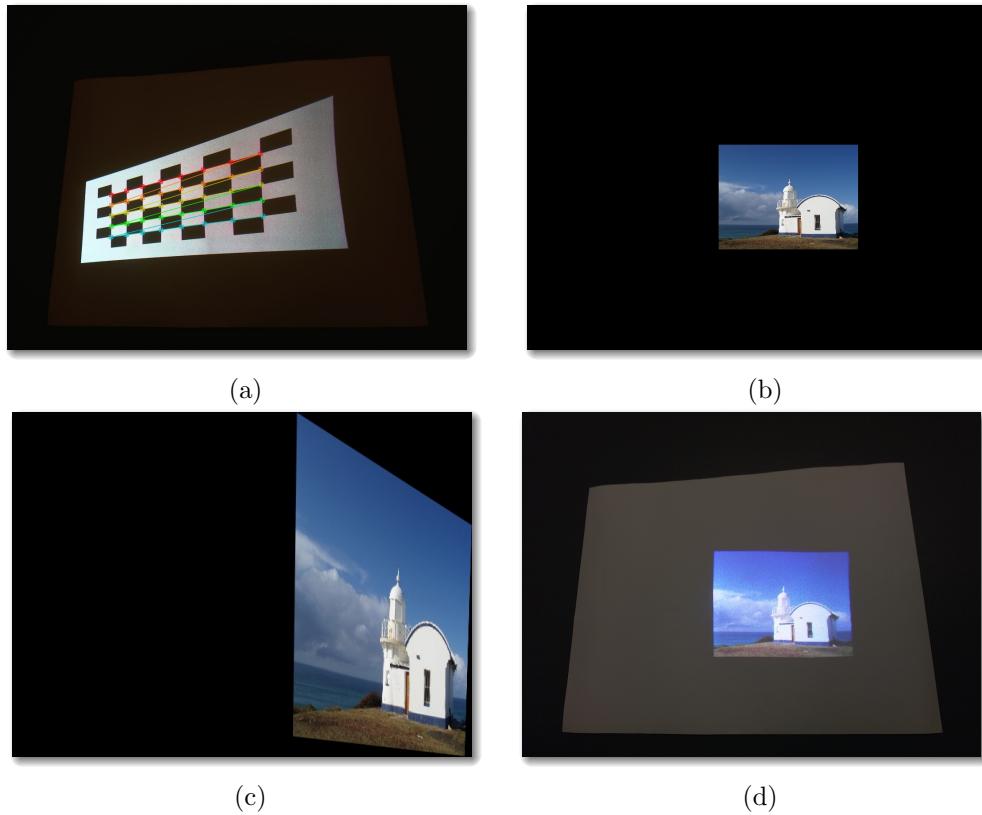


Figure 4.2.1: Screenshots of the program: (a) Detected chessboard corners. (b) Prepared pre-anamorph. (c) Anamorphic image after warping. (d) Anamorphic image seen correctly by the camera.

The modified algorithm is outlined below:

1. Project a chessboard pattern.
2. Find the inner corners of the chessboard in the projector and camera image.

3. Estimate the homography H_{PC} using the least squares method.
4. Prepare (resize and reposition) the pre-anamorph.
5. Generate the anamorph by warping the pre-anamorph with the homography

$$H_{CP} = H_{PC}^{-1}.$$

The result of this algorithm is compiled in Figure 4.2.1 and explained in detail later in this section.

4.2.1 Modified Set-up

We modify our setup to make it possible to find the homographies mapping to the projection surface. The Basic Model in Section 4.1 does not compute any homography involving the projection surface, so there is no need for the projection surface to have any detectable points. In fact, as seen in some camera captures of that model, for example Figures 4.1.3a and 4.1.3d, the projection surface is simply a large blank planar surface, and it is desirable that the the surface does not have any distinguishable features as they could be mistaken by the program for the corners of the projected rectangle that we need to detect. However, the new setup introduced in this section requires that the projection surface has some detectable features to facilitate the process of finding the homography mapping to it. We add a rectangular projection surface, on which we project images with the projector. We use a large rectangular piece of paper attached to a wall. The entire projection surface is then captured by the camera, so that the corners of the projection surface can be detected in the camera image. To find the homography H_{SC} from the projection surface to the camera image, we use the same homography estimation method as the one used to estimate the projector-camera homography H_{PC} .

4.2.2 Detecting the Projection Screen

We use the same good features to track algorithm to detect the corners of the projection screen. Unlike the projection (light from the projector), the screen is a solid object and its color appears stable in the camera, which means that the difference in color intensity between the projection screen and the background is more easily detectable than that between the projection and the background. Therefore the Good Features to Track algorithm, described in Section 3.3.2, is relevant as it provides a convenient way to detect exactly four corners of the projection surface. The detected corners are presented in Figure 4.2.2.

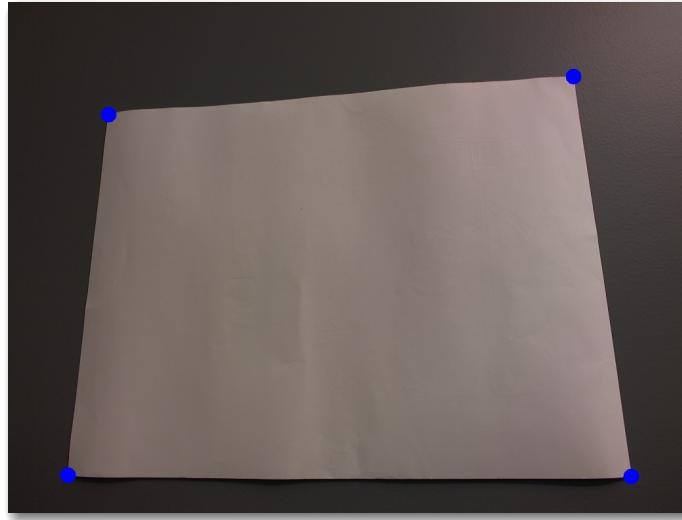


Figure 4.2.2: Detected projection screen corners marked with blue dots.

These corners in the camera image correspond to the corners of the projection surface in the real world. Using these four point correspondences, we can find the homography H_{SC} by using the estimation method described in Subsection 3.2.1, where the real-world corners of the projection surface assume the role of the projector points in the estimation method, and the detected corners of the surface in the camera image are the output points..

4.2.3 Homography Estimation with Chessboard Corner Detection

The Basic Model described in Section 4.1 uses a simple corner detection function to find the corners of a rectangular shape projected on a wall, as seen in the camera image (Figure 4.1.3a), and matches these corners with the corners of the original shape that was projected, as seen on the computer screen. These four corner correspondences are then used to find the homography H_{PC} . This method requires a specific light setting, and might not produce the desired results when the light is too bright as the projection might not be clearly visible in the camera image.

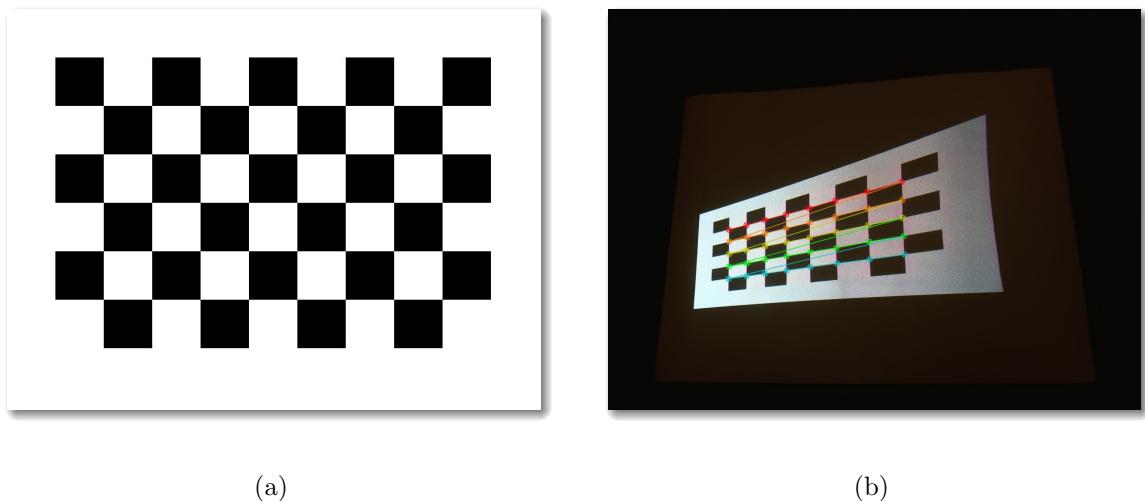


Figure 4.2.3: (a) Original chessboard corners in the projector image. (b) Detected chessboard corners in the camera image.

This subsection introduces an alternative way to obtain points in the projector and camera image that provide a better estimation of the homography H_{PC} . The major amendment is in the pattern projected. Instead of the simple rectangular pattern that contains only four detectable features, we project a 6×9 chessboard pattern, presented in Figure 4.2.3a, with $5 \cdot 8 = 40$ detectable inner corners. We choose this pattern because it is easily distinguishable from the projection screen. Notice that if we repeat the pattern detection process from the Basic Model (Section 4.1), the algorithm used to detect the projection

corners might not be able to distinguish between the corners of the projection and the projection surface. The chessboard provides an efficient solution because it can be easily detected regardless of other detectable objects surrounding it in the camera image, as shown in Figure 4.2.3b.

4.2.4 Repositioning and Resizing the Pre-anamorph

A major limitation of the Basic Model is the fact that the anamorph may not fit entirely within the projector image after warping. Recall that this phenomenon occurs because the entire projector image is mapped by H_{PC} into an area inside the camera image. It follows that the inverse homography H_{CP} maps the points within that projection area (inside the camera image) back to their positions in the projector image. However, the points in the camera image that lie outside of the projection area are mapped to new positions that do not fit inside the projector image, as illustrated in Figure 4.1.4.

We propose a method for resizing and repositioning the pre-anamorph before warping it to ensure that the resulting anamorph fits entirely inside the projector image and is of the largest size possible, described in the following steps:

1. Identify the four corners of the quadrilateral region of the projection area inside the camera image by applying H_{PC} to each of the four outer corners of the projection image. The detected projection is shown in Figure 4.2.4a.
2. Find the largest rectangle that fits inside the projection region in the camera image, as shown in Figure 4.2.4b. We explain this in more detail below.
3. Resize the original image so that it fits inside that rectangle, and place it in a blank (uniformly colored) camera image in a position that is aligned with the position of that rectangle (for example by aligning the top-left corner of the pre-anamorph with the top-left corner of that rectangle), as shown in Figure 4.2.4c.

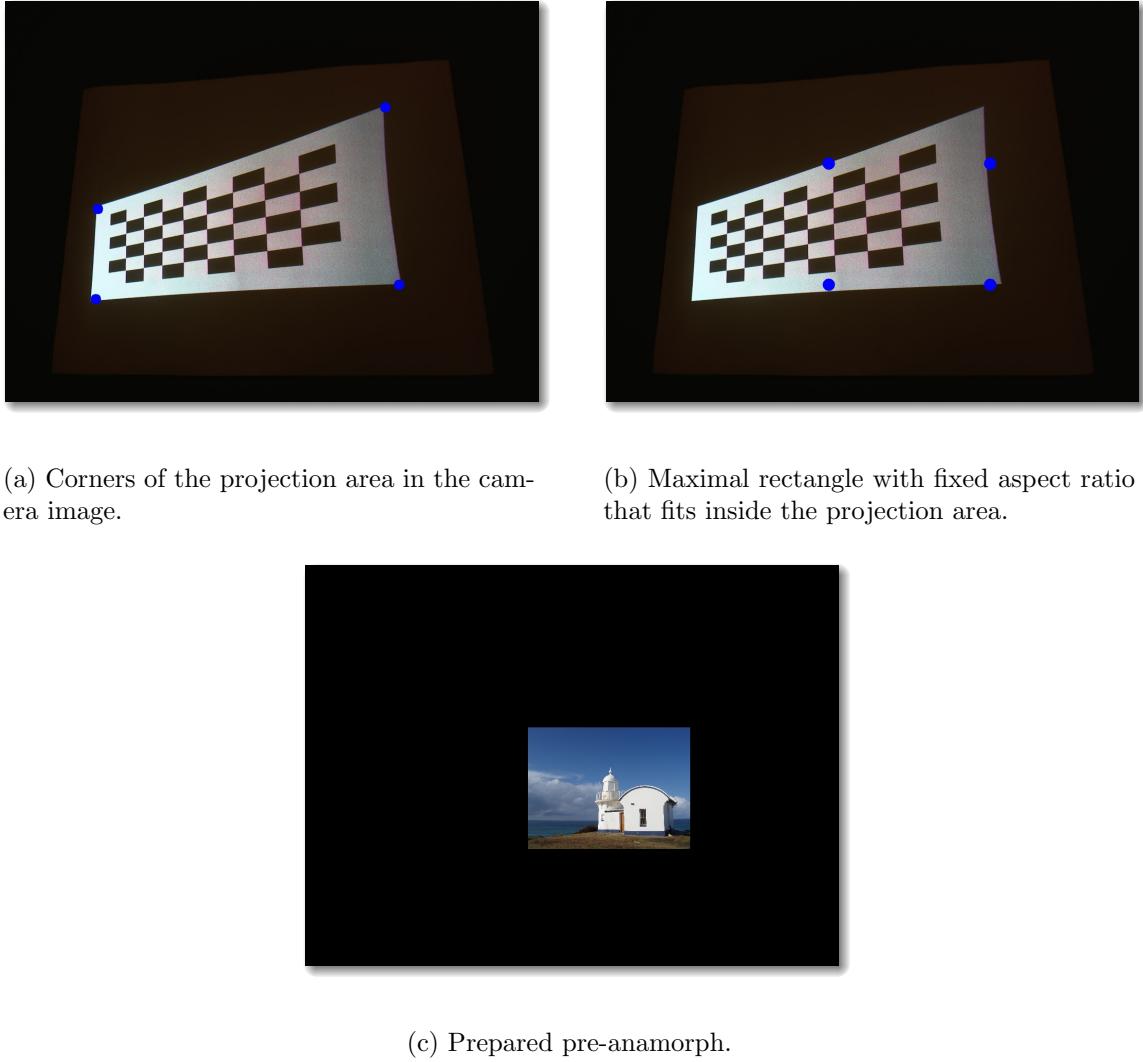


Figure 4.2.4: Screenshots of the program

After accomplishing these steps, the pre-anamorph is ready to be warped and is ensured to land inside the projector image.

It remains to explain the method for finding the largest rectangle inside a quadrilateral figure. Our method iterates through the pixels in the camera image and searches for the largest rectangle that fits inside the projection area. We improve the run-time of the algorithm by iterating only through the pixels that lie inside the projection area, before searching for the best rectangle, and we stop the iteration once the remaining area in the

camera image is smaller than the current largest rectangle. Additionally, we require that the rectangle has the same aspect ratio as the original image.

4.2.5 Challenges and Limitations

One minor issue we encounter when testing this model is that for some positions of the camera, the chessboard pattern is not detected correctly. One possible explanation is that the camera is at an angle (relative to the projection screen) such that the chessboard pattern appears too small, which causes the chessboard corners to be too crowded to be detected correctly. In addition, the projector light is not stable, which reduces the quality of the camera picture and makes it more difficult to accurately detect the chessboard corners. However, none of the tests performed have shown a noticeable deterioration in the quality of the anamorphic image. Because the number of chessboard corners is significantly larger than four, which is the minimum number of points required to establish a homography, the inaccuracy in detecting some of them is less likely to significantly affect the final result.

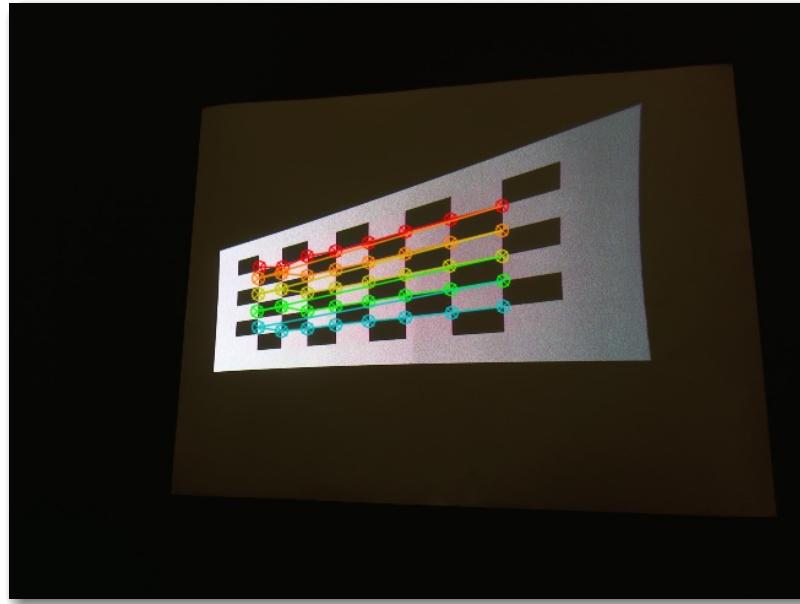


Figure 4.2.5: A scenario in which the program fails to detect the chessboard corners correctly.

4.3 Two-view Anamorphosis

By the definition of anamorphosis, there is only one perspective, or rather one ray, such that when a viewer looks at the anamorphic image from any point on that ray, the anamorphic image appears as the original unwarped image to the viewer. However, in practice there might be a number of viewers looking at the anamorphic image from several perspectives. In this section we address this issue and test a method for finding a transformation that minimizes the distortions for the viewers. Once again, we use cameras to represent viewers, and assume that the camera captures are what the viewers observe. Our goal is to produce a homography \hat{H} such that when \hat{H} is applied to an image, and the resulting anamorphic image is projected onto the projection surface, each camera capture of the projected image is “as close as possible” to the original unwarped image. Strictly following the definition of anamorphosis, the image transformed with that homography is not an anamorphosis for any of the cameras, because an anamorphosis can satisfy only one perspective. Rather than generating an image that is an anamorphosis for all the viewers, which is theoretically impossible, we aim to generate an image that is “as close as possible” to the anamorphosis for each viewer. Although the result of this program is not an anamorphic image, for consistency, we continue to use the terms “pre-anamorph” and “anamorph” to refer to the unwarped image and the final warped image that is projected, respectively.

The first question we need to answer is: how do we define “closeness” between the desired image and the actual image we obtain? In other words, what is the error? One way to define it is in terms of distance, which in this scenario means the distance between the same pixel in the two images in the x and y direction. We take the sum of the squared distances at each pixel as a measure of the error between the obtained and target image. Based on this definition, our reformulated goal is to find a homography such that when

it is applied to an image, and the image is projected on the projection surface, and then captured by the two cameras, the sum of the errors between the captures and the desired images is minimized. This problem is different from the previous two models in Sections 4.1 and 4.2 in that we need to minimize the error for both camera images, which makes the estimation of the optimal homography to warp the original image to the projector image significantly more complex.

The inclusion of the second camera requires that we introduce some changes to the program. We present the updated outline of the algorithm to generate an optimal image for two viewers:

1. Project a chessboard pattern.
2. Find the inner corners of the chessboard in Camera 1 and Camera 2 images.
3. Estimate the projector-camera homography for each camera, using the same method as in the Improved Model in Section 4.2.
4. Choose a target chessboard pattern for each camera.
5. Estimate the homography \hat{H} and its inverse \hat{H}^{-1} that map between the original image and the projector image, and minimize the difference in the observed chessboard corners and the target chessboard corners for each camera. The new homography notation is explained in the following Subsection 4.3.1.
6. Prepare (resize and reposition) the pre-anamorph.
7. Generate the anamorph by warping the pre-anamorph with the homography \hat{H}^{-1} .

4.3.1 Two-view Setup

The equipment needed for this setup is a projector, and projection surface, and two cameras. As in the previous models, we require that the projector and the cameras are directed towards the projection screen, and that the images projected land inside the projection

screen, and the cameras capture the entire projection screen. This setup is visualized in Figure 4.3.1.

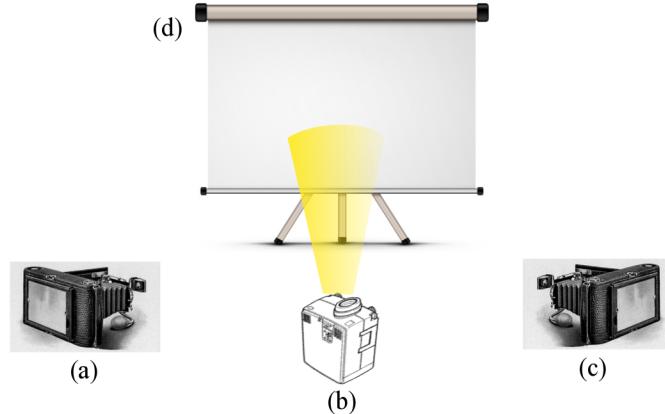


Figure 4.3.1: The new setup: (a) Camera 1 (b) Projector (c) Camera 2 (d) Projection surface.

Along with the changed setup, we also introduce a new notation of the homographies for this particular two view model, as presented in Figure 4.3.2. We denote the homography from the projector image to the first camera image as A , and the homography from the projector image to the second camera image as B . A and B correspond to the projector-camera homography H_{PC} from the one view models discussed earlier. We introduce a new homography \hat{H} from the projector image to the original image. It follows that the inverse homography \hat{H}^{-1} maps from the original image to the projector image, and that $A\hat{H}^{-1}$ and $B\hat{H}^{-1}$ map from the original image to the first and second camera image respectively. Recall that the previous one-view models use the homography H_{CP} to warp the pre-anamorph into the anamorph. This method stems from the valid assumption that the pre-anamorph is the target camera image, and the anamorph is the projector image, so the anamorph can be created by warping the target camera image into the projector image. However, this approach is not applicable to this model because the homography used to warp the pre-anamorph into the anamorph is neither A^{-1} (the camera-projector homography for camera 1) nor B^{-1} (the camera-projector homography for camera 1). In

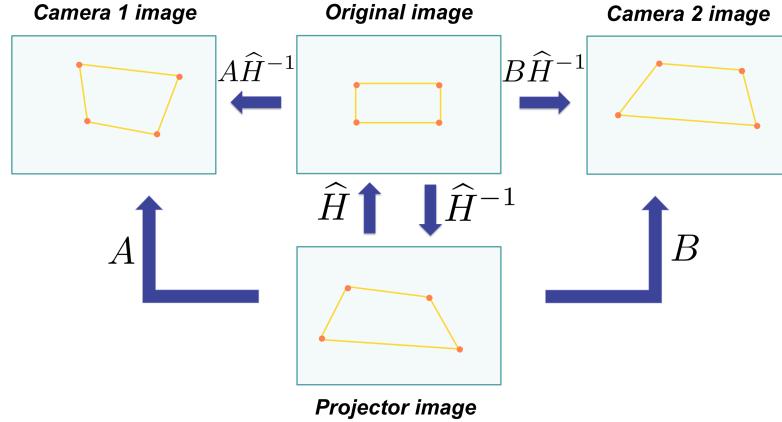


Figure 4.3.2: The new setup and notation for the homographies.

fact, if either A^{-1} or B^{-1} is used to create the anamorph, the result will be optimized for only one of the cameras. Therefore, this model requires a new homography \hat{H} to optimize the projected image for both camera views. The primary task in this model is to estimate the best homography \hat{H} as we need it to warp the original image (pre-anamorph) into the optimal anamorphic image, which is then projected.

4.3.2 Target Camera Images

The steps 1 through 3 outlined at the beginning of this section give an estimate of the homographies A and B using the estimation method described in the previous one-view models. It remains to determine the homography \hat{H} . Before we introduce the new method for homography estimation, we reiterate the approach used in the previous models to explain why it does not apply in the current two-view model, which sheds some light on the modifications we need to make to adapt the previous method to this scenario.

In the Improved Model approach, we project a chessboard pattern and detect the transformed chessboard corners in the camera image, as explained in Subsection 4.2.3. We then investigate the relationship between the original chessboard corners (in the projector image) and those detected in the camera image, and find the best homography H_{PC} that transforms the original chessboard points to the camera points. The reason why this

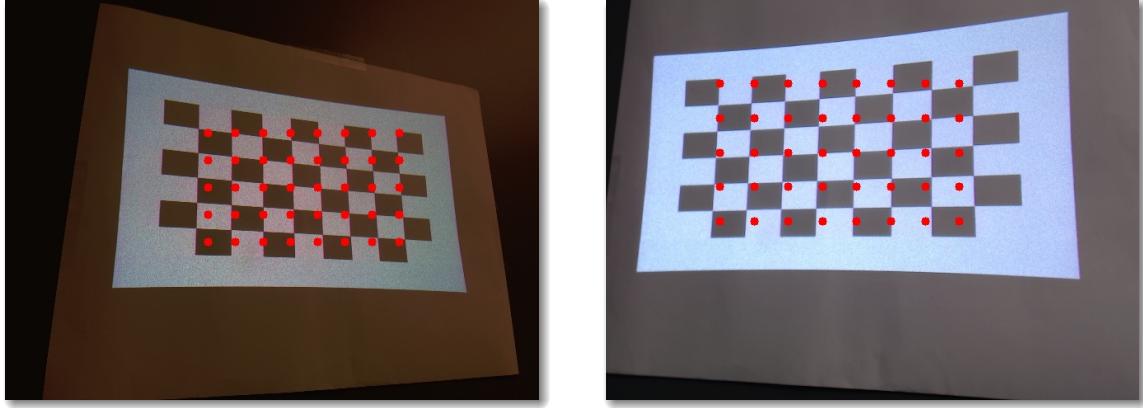
method can only be used to estimate the homographies A and B , but cannot be used to estimate the homography \hat{H} is that the previous approach considers the relationship of points between the projector and camera image, while \hat{H} relates points between the projector image and the original image (or the pre-anamorph).

To address this issue, we introduce a new estimation approach, which allows us to solve for the best homographies AH^{-1} and BH^{-1} between the original image and the camera image. Because A and B are already known to us from the earlier steps of the program, we effectively solve for H^{-1} . To do this, we need a set of points in the original image, and a corresponding set of points in each camera image. The points on the original image are simply the original chessboard corners. The target points on each camera image are the same as the original chessboard points, up to scale and translation. This follows from the reasoning that when the original image is the pre-anamorph (i.e. the picture that we use to generate the final anamorphic image), our goal is for the cameras to see the same picture as the original image, up to scale and translation.

To account for the location of the actual projection in each camera image, we find the new chessboard corners by performing a least squares minimization on the detected chessboard corners. In addition, we impose the following constraints on the new chessboard pattern:

1. The number of rows and columns in the new pattern is the same as that of the original chessboard pattern.
2. The chessboard corners are equally spaced.
3. The grid lines of the new pattern are parallel to the x and y axes.

The result of this method is presented in Figure 4.3.3



(a) New chessboard pattern for Camera 1
shown with red dots.
(b) New chessboard pattern for Camera 2
shown with red dots.

Figure 4.3.3: New chessboard patterns.

4.3.3 Homography Estimation

Due to the changes in the setup, the homography estimation method used in the previous models no longer applies. Therefore, we introduce a homography estimation method designed specifically for this model. While this technique bears some similarities to the previous method in that we continue to use the least squares minimization to find the optimal homography \hat{H} , the function that we minimize on is significantly different. This subsection provides a detailed explanation of the new homography estimation using the homogeneous least squares for the two view model.

Let p_0, p_1, \dots, p_{n-1} be the chessboard corners in the original image. Let v_0, v_1, \dots, v_{n-1} and w_0, w_1, \dots, w_{n-1} be the target chessboard corners in Camera 1 and Camera 2 respectively. The homography that maps points from the original image to a camera image is $A\hat{H}^{-1}$ for Camera 1 and $B\hat{H}^{-1}$ for Camera 2, where the entries of \hat{H}^{-1} are the unknowns. Let $h_{00}, h_{01}, \dots, h_{22}$ be the entries of \hat{H}^{-1} so that

$$\hat{H}^{-1} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}.$$

Without loss of generality, we assume that $h_{22} = 1$. By the homogeneous least squares described in Subsection 3.2.2, the error function E can be defined as:

$$\begin{aligned} E(h_{00}, h_{01}, \dots, h_{21}, \lambda_0, \lambda_1, \dots, \lambda_{n-1}, \mu_0, \mu_1, \dots, \mu_{n-1}) &= \sum_{k=0}^{n-1} \left(\|A\hat{H}^{-1}p_k - \lambda_k v_k\|^2\right) \\ &\quad + \sum_{k=0}^{n-1} \left(\|B\hat{H}^{-1}p_k - \mu_k w_k\|^2\right) \end{aligned}$$

for some scaling variables λ_k corresponding to each target point v_k in Camera 1, and μ_k corresponding to each target point w_k in Camera 2. The scaling factors account for the fact that any multiple of a homogeneous point corresponds to the same point, and so taking the difference between $A\hat{H}^{-1}p_k$ and a multiple of v_k , and the difference between $B\hat{H}^{-1}p_k$ and a multiple of w_k gives a better estimate of the actual distance between these pairs of points than, for example taking the difference of $A\hat{H}^{-1}p_k$ and unscaled v_k .

Observe that $\|A\hat{H}^{-1}p_k - \lambda_k v_k\|$ and $\|B\hat{H}^{-1}p_k - \mu_k w_k\|$ are of similar structure, which allows us to rewrite the error function in an equivalent but simpler form:

$$E(X_{00}, X_{01}, \dots, X_{21}, \sigma_0, \sigma_1, \dots, \sigma_{2n-1}) = \sum_{k=0}^{2n-1} \|A_k X b_k - \sigma_k c_k\|^2$$

where A_k is A or B , X is \hat{H}^{-1} , b_k is p_{k-n} , σ_k is λ_k or μ_{k-n} , and c_k is v_k or w_{k-n} , depending on the value of k , as shown in Table 4.3.1.

	A_k	X	b_k	σ_k	c_k
$0 \leq k \leq n-1$	A	\hat{H}^{-1}	p_k	λ_k	v_k
$n \leq k \leq 2n-1$	B	\hat{H}^{-1}	p_{k-n}	μ_{k-n}	w_{k-n}

Table 4.3.1: The meaning of new variables A_k , X , b_k , σ_k and c_k .

The goal of this section is to find $(h_{00}, h_{01}, \dots, h_{21}, \sigma_0, \sigma_1, \dots, \sigma_{2n-1})$ that minimize the error function. As explained in Section 3.1, we find partial derivatives of E with respect to each variable, and solve a system of linear equations in which each partial derivative is zero. For simplicity, we perform the differentiation on the simple form of the error function. The

solution to that system of equations is the vector $(X_{00}, X_{01}, \dots, X_{21}, \sigma_0, \sigma_1, \dots, \sigma_{2n-1})$. In particular, the values $X_{00}, X_{01}, \dots, X_{21}$ in the solution are the entries of the homography matrix \widehat{H}^{-1} .

The partial derivatives take two different forms depending on the variable with respect to which we differentiate. We distinguish these two types of variables: X_{rs} where $0 \leq r, s \leq 2$, and σ_r where $0 \leq r \leq 2n - 1$.

The partial derivative of E with respect to X_{rs} for any r, s such that $0 \leq r, s \leq 2$ is:

$$\begin{aligned} & \frac{\partial}{\partial X_{rs}} \sum_{k=0}^{2n-1} \|A_k X b_k - \sigma_k c_k\|^2 \\ &= 2 \sum_k (A_k X b_k - \sigma_k c_k) \cdot (A_k E_{rs} b_k) \end{aligned} \quad (4.3.1)$$

$$= 2 \sum_k \sum_l (A_k X b_k - \sigma_k c_k)_l (A_k E_{rs} b_k)_l \quad (4.3.2)$$

$$= 2 \sum_k \sum_l ((A_k X b_k)_l - \sigma_k (c_k)_l) ((A_k)_{lr} (b_k)_s) \quad (4.3.3)$$

$$\begin{aligned} &= 2 \sum_k \sum_l \left(\left(\sum_{i,j} (A_k)_{li} X_{ij} (b_k)_j \right) - \sigma_k (c_k)_l \right) ((A_k)_{lr} (b_k)_s) \\ &= 2 \left(\sum_{i,j} \sum_k \sum_l (A_k)_{li} (b_k)_j (A_k)_{lr} (b_k)_s X_{ij} - \sum_k \sum_l (c_k)_l (A_k)_{lr} (b_k)_s \sigma_k \right). \end{aligned} \quad (4.3.4)$$

The E_{rs} in lines (4.3.1) and (4.3.2) of the derivation above is defined as a matrix with one at the position rs , and zero everywhere else. For aesthetic purposes we skip the bounds on the sums in the derivation above, and explain them here. The index iterator l results from taking the dot product of $A_k X b_k - \sigma_k c_k$ and $A_k E_{rs} b_k$, each of which are vectors of length 3, therefore $0 \leq l \leq 2$. The index iterators i and j result from taking the l^{th} entry of the product of A_k , X and b_k . Since A_k and X are 3×3 matrices, and b_k is a vector of length 3, it follows that $0 \leq i, j \leq 2$. From (4.3.1) to (4.3.2), we use the fact that the dot product of two vectors \vec{u} and \vec{v} is $\vec{u} \cdot \vec{v} = \sum_l u_l v_l$. From (4.3.2) to (4.3.3) we use the

following derivation:

$$\begin{aligned}
(A_k E_{rs} b_k)_l &= (A_k (E_{rs} b_k))_l \\
&= \left(A_k (0, \dots, (b_k)_s, \dots, 0)^T \right)_l, \text{ where } (b_k)_s \text{ is the } r^{\text{th}} \text{ entry} \\
&= (A_k)_{lr}(b_k)_s.
\end{aligned}$$

From (4.3.3) to (4.3.4) we use these two properties:

$$\begin{aligned}
(A\vec{v})_l &= \sum_i A_{li} v_i, \text{ and} \\
(AB\vec{v})_l &= \sum_i A_{li} (B\vec{v})_i = \sum_i \left(A_{li} \sum_j B_{ij} v_j \right) = \sum_{ij} A_{li} B_{ij} v_j.
\end{aligned}$$

We set this derivative to zero, and rearrange it so that the constant terms are on the right-hand side. Because $X_{22} = 1$, the term X_{22} and its coefficients are moved to the right-hand side:

$$\begin{aligned}
2 \left(\sum_{i,j} \sum_k \sum_l (A_k)_{li} (b_k)_j (A_k)_{lr} (b_k)_s X_{ij} - \sum_k \sum_l (c_k)_l (A_k)_{lr} (b_k)_s \sigma_k \right) &= 0 \\
\sum_{i,j} \sum_k \sum_l (A_k)_{li} (b_k)_j (A_k)_{lr} (b_k)_s X_{ij} - \sum_k \sum_l (c_k)_l (A_k)_{lr} (b_k)_s \sigma_k &= 0 \\
\sum_{(i,j) \neq (2,2)} \sum_k \sum_l (A_k)_{li} (b_k)_j (A_k)_{lr} (b_k)_s X_{ij} - \sum_k \sum_l (c_k)_l (A_k)_{lr} (b_k)_s \sigma_k &= \sum_k \sum_l (A_k)_{l2} (b_k)_2 (A_k)_{lr} (b_k)_s. \quad (4.3.5)
\end{aligned}$$

Converting the equation (4.3.5) back to the notation used in our initial setup of this model, we have the following equation:

$$\begin{aligned}
&\sum_{(i,j) \neq (2,2)} \left[\sum_k \sum_l A_{li}(p_k)_j A_{lr}(p_k)_s + B_{li}(p_k)_j B_{lr}(p_k)_s \right] h_{ij} \\
&- \sum_k \left[\sum_l (v_k)_l A_{lr}(p_k)_s \right] \lambda_k \\
&- \sum_k \left[\sum_l (w_k)_l B_{lr}(p_k)_s \right] \mu_k \\
&= - \sum_k \sum_l [A_{l2}(p_k)_2 A_{lr}(p_k)_s + B_{l2}(p_k)_2 B_{lr}(p_k)_s].
\end{aligned}$$

The partial derivative with respect to σ_r for any $0 \leq r \leq 2n - 1$ is:

$$\begin{aligned}
& \frac{\partial}{\partial \sigma_r} \sum_{k=0}^{2n-1} \|A_k X b_k - \sigma_k c_k\|^2 \\
&= \frac{\partial}{\partial \sigma_r} \|A_r X b_r - \sigma_r c_r\|^2 \\
&= -2 (A_r X b_r - \sigma_r c_r) \cdot c_m \\
&= -2 \sum_l (A_r X b_r - \sigma_r c_r)_l (c_r)_l \\
&= -2 \sum_l \left(\left(\sum_{i,j} (A_r)_{li} X_{ij} (b_r)_j \right) - \sigma_r (c_r)_l \right) (c_r)_l \\
&= -2 \left(\sum_{i,j} \sum_l (A_r)_{li} (b_r)_j (c_r)_l X_{ij} - \sum_l (c_r)_l^2 \sigma_r \right).
\end{aligned}$$

Notice that in the process of taking the derivative, the summation on k disappeared.

This is because $\frac{\partial}{\partial \sigma_r} \|A_k X b_k - \sigma_k c_k\|^2 = 0$ for all $k \neq r$. We explain the limits on the summations above: l is the iterator in the dot product of $A_r X b_r - \sigma_r c_r$ and c_m , both of which are vectors of length 3, so $0 \leq l \leq 2$; i and j are iterators in taking the l^{th} entry of the product of A_k , X and b_k , and since A_k and X are 3×3 matrices and b_k is a vector of length 3, it follows that $0 \leq i, j \leq 2$.

Again, we set the derivative to zero and bring the constant terms to the right-hand side:

$$\begin{aligned}
& -2 \left(\sum_{i,j} \sum_l (A_r)_{li} (b_r)_j (c_r)_l X_{ij} - \sum_l (c_r)_l^2 \sigma_r \right) = 0 \\
& \sum_{i,j} \sum_l (A_r)_{li} (b_r)_j (c_r)_l X_{ij} - \sum_l (c_r)_l^2 \sigma_r = 0 \\
& \sum_{(i,j) \neq (2,2)} \sum_l (A_r)_{li} (b_r)_j (c_r)_l X_{ij} - \sum_l \sigma_r (c_r)_l^2 = \sum_l (A_r)_{l2} (b_r)_2 (c_r)_l. \quad (4.3.6)
\end{aligned}$$

Converting to the notation used in our initial setup of this model, we obtain two equations depending on the value r . When $0 \leq r \leq n - 1$, $\sigma_r = \lambda_r$, and the equation (4.3.6) is

$$\sum_{(i,j) \neq (2,2)} \left[\sum_l A_{li}(p_r)_j (v_r)_l \right] h_{ij} - \left[\sum_l (v_r)_l^2 \right] \lambda_r = - \sum_l A_{l2}(p_r)_2 (v_r)_l.$$

When $n \leq r \leq 2n - 1$, $\sigma_r = \mu_{r-n}$ and the equation (4.3.6) is

$$\sum_{(i,j) \neq (2,2)} \left[\sum_l B_{li} (p_{r-n})_j (w_{r-n})_l \right] h_{ij} - \left[\sum_l (w_{r-n})_l^2 \right] \mu_r = - \sum_l B_{l2} (p_{r-n})_2 (w_{r-n})_l.$$

We solve the resulting system of equations for the unknowns $h_{00}, h_{01}, \dots, h_{21}, \lambda_0, \lambda_1, \dots, \lambda_{n-1}, \mu_0, \mu_1, \dots, \mu_{n-1}$, where $h_{00}, h_{01}, \dots, h_{21}$ and 1 are the entries of the homography \hat{H}^{-1} .

4.3.4 Generating the Anamorphic Image

To generate the anamorphic image, we perform the repositioning and resizing steps described in Subsection 4.2.4 in the Improved Model. We then apply the \hat{H}^{-1} to the prepared pre-anamorph, which lies on the original image, to transform it into the anamorph on the projector image.

Our initial attempts did not provide expected results prompting us to identify some potential shortcomings in our model. We tested the algorithm for homography estimation using the homogeneous least squares method on specific parameters to verify if the algorithm is correct. We compare the obtained test results with a parallel computation in Mathematica. The results of the tests suggest that the implementation works for parameters that can be expressed with few decimal digits, but does not give the expected solution for parameter with many decimal digits. This finding led us to consider the possibility of encountering ill-conditioned matrices, which are matrices in which the difference in between the smallest and largest entries is large, which causes the program to lose some accuracy when performing Gaussian elimination on those numbers. Further tests confirmed this conjecture as the condition number was between 4 and 10, which classifies the coefficient matrix in our model as ill-conditioned (a well-conditioned matrix has the condition number approximately 1).

We attempted to address this problem by using the singular matrix decomposition. Despite some improvements in accuracy when calculating the homography on the test

parameters, the problem of generating the anamorphic image remains unresolved. Due to the time constraints, we hope to look further into this issue in future research.

5

Future Work

In this project we explore the mathematical concepts behind perspective planar anamorphosis, and develop models for generating the anamorphic images on planar surfaces for single and two-view model using a simple projector-camera system. We design and implement a homography estimation method using homogeneous least squares for the two-view model. However, we encounter some challenges that need to be addressed in the future. Some improvements include developing more accurate feature detection techniques, and solving the ill-conditioned matrices problem, which might improve the results of the two-view model.

This project may serve as a basis for other research directions. One adaptation of the findings of this work is in keystone correction, which will take advantage of the modified setup in the improved model in Section 4.2. This project could be extended by including a real-time tracking of the viewer's position to create a dynamic anamorphosis system. Another direction that could be taken is investigating anamorphosis for arbitrary surfaces, which could be done for example with adaptive structured lighting techniques. The topics studied in this project have a potential for other practical applications such as enhanc-

ing viewers' experience by embedding projected images in performances such as plays or dance. To expand the scope of the applications of anamorphosis in performances, one could develop a model of anamorphosis for more than two viewers, which can ultimately be used to optimize the projection for a large audience. One way to address the large audience scenario is to build a synchronized system of multiple projectors, which can potentially generate more powerful and engaging images. A paper by Elodie presents another interesting approach to the subject of anamorphosis by considering the Renaissance art and investigating the way artists of that time dealt with perspective. [4] This approach may shed some light on how humans perceive the world, and cast some doubt on whether strictly following the perspective principles yields an image that is convincing to the human eye.

Bibliography

- [1] R. E. Allardice, *The Barycentric Calculus of Möbius*, Proceedings of the Edinburgh Mathematical Society **10**, 2–21.
- [2] Alessandro Brazzini and Carlo Colombo, *Computer vision for interactive skewed video projection*, Image Analysis and Processing–ICIAP 2005, 2005, pp. 139–146.
- [3] Wilhelm Burger and Mark J. Burge, *Digital image processing: an algorithmic introduction using Java*, Springer Science & Business Media, 2009.
- [4] Elodie Fourquet, *Learning about shadows from artists*, Proceedings of the Sixth international conference on Computational Aesthetics in Graphics, Visualization and Imaging, 2010, pp. 107–114.
- [5] Chris Harris and Mike Stephens, *A combined corner and edge detector.*, Alvey vision conference **15** (1988), 50.
- [6] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2003.
- [7] J. L. Hunt, BG Nickel, and Christian Gigault, *Anamorphic images*, American Journal of Physics **68** (2000), no. 3, 232–237.
- [8] Jean François Niceron, *Perspective curieuse*, JD Puis, 1992.
- [9] Robert Ravnik, Borut Batagelj, Bojan Kverh, and Franc Solina, *Dynamic Anamorphosis as a Special, Computer-Generated User Interface*, Interacting with Computers (2013), iwt027.
- [10] David Eugene Smith and Mansfield Merriman, *History of modern mathematics*, J. Wiley & Sons, 1896.
- [11] Franc Solina and Borut Batagelj, *Dynamic anamorphosis* (2007).
- [12] Rahul Sukthankar, *Smarter presentations: Exploiting homography in camera-projector systems*, Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference **1** (2001), 247–253.

- [13] Richard Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.
- [14] Carlo Tomasi and Jianbo Shi, *Good features to track*, CVPR94 **600** (1994), 593–593.
- [15] *OpenCV documentation*, <http://docs.opencv.org/>.