Lighting: A Systems Approach

A Thesis
Presented to
The Established Interdisciplinary Committee for
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Gabriel Howland

May 2025

Approved for the Committee
(Computer Science and Theatre)

Peter Ksander	Jim Fix

Acknowledgements

I want to thank a few people. It's gonna happen later.

List of Abbreviations

You can always change the way your abbreviations are formatted. Play around with it yourself, use tables, or come to CUS if you'd like to change the way it looks. You can also completely remove this chapter if you have no need for a list of abbreviations. Here is an example of what this could look like:

DMX	Digital Multiplex
ACN	Advanced Control Network
sACN	Scanning ACN
DLT	Direct Linear Transform
SVD	Singlular Value Decomposition

Contents

List of Figures

Abstract

This thesis studies the development of lighting control throughout history, and attempts to contribute to the next step in controlling lights. In this, I recap the lighting control techniques from the days of candles to DMX. I propose a stereoscopic vision system to track a performer in a theatrical space, and cover the process to creating such a software. Part of that includes an overview of photogrammetry, and the process for calibrating cameras, locating objects in three-dimentional space, and positioning a light in said space.

Introduction

This thesis explores lighting control, specifically for live performance. It will take a look at how lighting was originally controlled manually, how the technology has advanced to today through the use of time coding, and proposes a system for control that is reactive to a performer. Lighting control has advanced at a breakneck speed over the past half-century as the world entered a digital age. Where lighting control rooms were packed with levers and room-scale dimming racks now sit lighting desks, or even just a laptop. As the technology for controlling lights has advanced, the lighting design for live performances has gotten more complicated, while being largely prerecorded. Thus, while a designer's vision is accomplished in collaboration with the director and performers, it stays static barring catastrophe, which introduces interesting problems. Consider the actor who moves in tandem to a moving light. This is usually accomplished through hours of rehearsal, and fine-tuning movements in order to keep pace with the light. However, if the actor in a particular showing wanted to modify a movement, either in the route or in the speed, they would be constrained by the lighting.

This thesis attempts to alleviate those constraints by creating a lighting control scheme that tracks a performer through a theatrical space. It is a reactive scheme that hopes to meet a few criteria: 1) The scheme must be lightweight; 2) The scheme must be able to run using off-the-shelf components and computing resources; 3) The scheme must be open source; 4) The scheme should run on top of existing theater infrastructure. It is worth mentioning that this concept is not revolutionary. There

are numerous choices for high-end performer tracking offered by the likes of Cast Group's BLACKTRAX, Follow-Me, and zactrack. However, these systems are often closed-source, prohibitively expensive, hardware intensive, or otherwise difficult for small productions to access. The software developed during this thesis hopes to try to make it easier to access, which is why the scheme has these criteria.

This thesis is spread across three chapters. The first is a literature review of the theory and practice behind lighting control. It will start at the very beginning, mechanical ropes and pulleys, hiding and revealing gas lamps, all the way to the modern control systems that run live performance today. It will look at how the jump from analog to digital control marked a shift in the complexity of lighting design. It will explore how this technology differs over productions of different sizes with the theory that that technology often "trickles down" from high-end productions (concert tours, sporting events, Broadway, etc.) to low-end, and most technology is manufactured for the arenas, concert venues, and other performance spaces who can pay the premium.

The second chapter looks at this thesis with a different perspective, namely through the lens of photogrammetry (or the study of cameras). It will be an exposition of the different computational problems that come with tracking a perfomer and what was done to solve them. These problems are, including, but not limited to: calibrating cameras, tracking the performer, and positioning the moving light.

The third chapter will explore the culmination of the work done in the previous two chapters, and the stumbling block I hit when preparing to present the software in a lighting design for the Dance thesis of Beier (Belle) Li. This covers the design process of Li's thesis, the issues relating to scale that I ran into, and what the next steps were in creating a working and presentable demo of the software.

Chapter 1

The Problem

1.1 A History of Lighting Control

We begin in the 16th century, in the "cradle of stage lighting" that was the Italian Renaissance. Here lies the earliest records and examples of a "standard practice" for lighting a theatre. When looking at the compiled histories of lighting and its role in theatre, authors agree that the distinction of lighting from scenic elements took its hold here. Richard Pilbrow's history of lighting technology in Stage Lighting Design indicates that there is evidence of paint and reflective costumes compensating for the less visually interesting sunlight and torchlight used previously: "A stage direction of 1501 in the Mona Passion reads, 'See to it that the painter goes to Paradise to paint Raphael's face red.' In the same play, Christ is described with gold hands and feet, and 'let there be a big sun behind him.'" Max Keller acknowledges that there was use of artificial lighting in Greek and Roman theatre, but "nothing has been recorded or passed down to us about theatrical lighting as we understand it."

The first recorded lighting control came at the turn of the 17th century. Nicola Sabbattini, an Italian architect, outlined a rudimentary system of dimming light through tin cans suspended over candles or lamps. He describes it as follows: "When

it is desired to darken the whole stage in a moment, this method is used: as many cans of soldered tin are made as there are lamps to be darkened. [This] done, you adjust each cylinder over its lamp [in] such a manner that by one motion on the side of the stage, the cylinders descend over the lamps and darken them." Wax and oil was expensive, and in order to be able to see performers, many were needed. Theatres opted to use thousands of candles or lamps positioned strategically to cast the best light on the performers. The process of lighting and maintaining the candles was a gargantuan task, requiring many technicians to aid. Sabbattini mentions this in his text: "Every care must be taken to [light the candles] as quickly as possible to avoid restlessness in the spectators who think this business is endless." Wicks needed to be trimmed, lamps refilled, and molten wax would sometimes fall upon the spectators.

The next jumps in technology appeared in the 19th century when gas-powered light entered the theatrical scene in 1803 with a demo in the Lyceum Theatre. Full adoption appeared in the Chestnut Theatre in 1816 and the Lyceum in 1817. Soon after, the gas table—a contraption with levers, valves, and piping designed to control the flow of gas to any location in the theatre—became produced by companies like Clemancon. These gas tables are considered the first "switchboard" for theatrical lighting, or elaborate control schemes that could modify the intensity of a light from a single location. These tables had pipes that would control sections of the theatre, like footlights, auditorium lights, proscenium lights, etc.

A byproduct of gas light was the discovery of burning quicklime with a mixture of hydrogen and oxygen. The resulting lime would create a bright white light source; when reflected, created a pure beam of light, allowing a performer to be hit with a dazzling light. This type of light was named after the quicklime used inside its body, and thus "limelight" was created in 1826. This style of lighting allowed for brighter, whiter light while still being controlled by the gas table. The directness of the light also allowed it to be turned into the first follow-spot.

Following the advent of the incandescent lamp in 1879, the Savoy Theatre in London fully converted from gas-light to electric light. Large theatre venues were quick to adopt the technology. After the Savoy's adoption of electric lighting, most large venues fully converted by 1887. This rapid conversion to electric lighting was due to the sheer improvements an incandescent bulb gave over gas. Gas light consumes oxygen, produces fumes, and lots of heat. Compared to gas, incandescent lamps produce less heat, and consume no oxygen creating a much nicer theatre going experience: "A Captain Shaw noted that the temperature in the grid of the Savoy was 68deg Fahrenheit, while at the Alhambra, lit with gaslight, it was 105deg." There was also a dramatic safety improvement from removing open flames from the theatrical spaces. During the Savoy's opening, proprietor Richard D'Oyly Carte "appeared before the curtain and demonstrated the new safety of electricity by smashing a lit lamp wrapped in muslin. This was greeted with tumultuous cheers."

The next hurdle to clear was how to control this newfangled electricity, and the answer was resistance. Multiple mediums were used: be it sand, water, or different amounts of copper; the goal was to introduce resistance to reduce the current passing through an incandescent filament. In its early introduction, dimmers took up lots of room; the gas tables—which could be operated by a single person—turned into room scale operations, humming with electricity. Lighting control was dominated in the 1930's by systems such as the Bordoni and Salani control systems, which were variable rate transformers. These were mechanical apparati, with interlocking gears that often required multiple technicians to manage it. A byproduct of these roomscale dimmers was the Grand Master and Master control arms. These were levers or wheels that could be locked to specific dimmers to allow all of them to dim at the same time, even if set at different levels.

Technological advancements progressed swiftly. The next step was to shrink the footprint of the control surface. This was originally done through hundreds of small

lines of wire called "tracker wire" which allowed the levers or switches to be placed closer together—and more importantly, further from the dimmers—into one large bank. Not twenty years later vacuum tubes and relays were all the rage, and the "light organ" became the first lighting console that was mechanically independent from the dimmers (opting to control the dimmers electrically rather than mechanically). These were consoles that controlled lighting at a single desk, not running about a room. Like the gas table previously, the job of lighting a theatrical space returned to just one operator. This cut down on technicians and costs. Newer consoles allowed for presets and, shortly after, memory. This was initially set up on punch cards, but as magnetic tape became prominent in computing, it too transitioned to theatre. Once memory was introduced, technicians no longer had to scramble to set each scene, and designers could just load scenes from memory; the console could recreate it with minimal effort.

In 1986, the US Institute for Theatre Technology (USITT) made the jump to digital with a signaling protocol called Digital Multiplex (DMX). With it, (and its revision in the 1990), lighting control devices continued to shrink. The three decades following the genesis of DMX paved the way for new protocols, more advanced lanterns, and a full commitment to the world of saving and loading cues. The Musical Instrument Digital Interface (MIDI) which was created in 1983, was adapted as a control protocol to allow remote activation of lighting cues, often in time with sound or video cues. MIDI then had to contend with Open Sound Control (OSC), a control protocol created in 2002, which allowed specificity in control messages. DMX continues to be the dominant communication protocol, with spaces taking multiple universes (or groupings of 512 addresses) as automated lights require more.

1.2 Taking the "Live" out of "Liveness"

Throughout this history, the number of technicians required to run a show fluctuates. For example, back in the candle and oil lit theatres of the Italian Renaissance, Sabbatini recommends a large number of technicians to work together to simultaneously light the candles at the beginning of the show. For the chandeliers that sat above the house, he specifies that three technicians tend each fixture. Similarly, when operating gas lights, the Lyceum Theatre employed a "[large number of men] to look after the gas, to light and turn it off as required." It is with the gas table that records show that a smaller number of operators are required to run a show—as few as one was needed to operate the table because control was made available in one location. The need for technicians grew again with the advent of electrical dimming, as the size of the control surface expanded to room-scale, then soon decreased as the control surfaces consolidated into a single operator at a lighting desk. As modern technology continues to improve, there is not a growth in need for technicians. In fact, as the advancements in control moved from hardware to software, there is less of a need for operators as a whole.

A present factor in modern lighting—and overall show—control that evolved out of the ability to digitally record cues is timecoding. The concept of timecoding is that there is a global clock running through a performance and certain lights activate after a set amount of time. Time- coding is a technique that often plays nice with sound and video cues, allowing lights to match a prerecorded video clip or sound file down to the millisecond. In this modern age, timecoding can affect all aspects of a performance, including the human performers. This begins to present a problem if a performer has any deviation in their performance: "A time based system is less forgiving to human performers. If, for example, the [cue was] triggered at 4 minutes and 35 seconds into every performance, the performer would be out of luck if their performance varied much." Timecoding is effective as a synchronization technique,

but can be a disservice to performers because the computer will not stop without human intervention.

This presents a problem, and one that is the foundation of The Problem that exists in my thesis. As this technology has advanced, the need for a human operator—or multiple operators—decreases. Throughout all the previously referenced texts, a through line was present: the operator is intrinsically tied to the performance.

In the introduction of David Haye's Light On The Subject, Peter Brook mentions the process cuing a production of Hamlet in the Moscow Art Theatre. The lighting designer, Joe Davis, had just finished his 100th cue. He instructed the operators to go back to the first cue to which they registered confusion.

> In the Moscow Art Theatre, a lighting plot was unknown. The electricians would be present at every rehearsal, which unlike our miserable four weeks would often last two years. They ended up knowing the play as well as the actors, and slowly built up the lighting stroke by stroke, day by day. When the performers came, they did not work by cues, they lived the lighting changes as the actor lived his entrances, his exits and his changing moods.

While two years is a long time to produce and create a show, the idea that the operators were as ingrained with the performance as the actors themselves persists internationally. The concept of "lighting rehearsals" (akin to dry tech now) was conceived as a rehearsal entirely for the operators to familiarize themselves with the performance without actors present.

Nowadays, the timecoding and networking of lighting with other media elements continues to reduce the requirement of a technician to even operate the lighting console. This troubles the concept of liveness, which is one of the main draws of theatre. Part of that concept of liveness expresses itself as a fundamental difference between a repeatable and perfect show, and the continual effort to reach said show. The

rehearsal process, for actors and operators, progresses towards a perfect but there always runs the risk of variability. Director Jordan Tannahill says as such in his essay, "Why Live: A Question for 21st Century Theatre": "We continue to rehearse and perform shows with the aim of reproducing the same event—the same text, the same ebbs and flows of laughter and pathos, the same moments of revelation—night after night. In other words, we often aim to create an inert, knowable, and replicable entity." It is this aim that aids liveness, and timecoding runs counter to this. Theodore Fuchs expresses displeasure at the use of digital recording of cues, comparing it to "a pianola used in place of a pianist in an orchestra of live musicians."

The issue is money. Hiring operators is usually more expensive than paying for software. This monetary incentive is present in the upgrades to lighting hardware as well. Electricity was cheaper than gas which was cheaper than candles. Liveness is literally being "priced out."

1.3 The Solution (A Solution?)

The direction that software is headed is in allowing performers to be less restricted by timecoding. Specifically, the more recent advances in software appear in the manipulation of automated lighting.

When I refer to automated lighting, what I am referring to is a lighting fixture (or luminaire) with the ability to change the direction of the beam through the use of motors. Also known as moving lights, movers, or "intelligent" lights, these are controlled through a series of computer instructions, most often through the DMX or sACN protocols. Movers have a variety of sizes and capabilities from the Rosco I-Cue™—an attachment placed on the end of an instrument with a moving mirror—to the monstrous Vari-Lite VL3000, a lamp mounted on a two-axis system that allows for the whole unit to pan and tilt. These moving lights are expensive, with price

ranges starting in the thousands.

Movers are excellent at a couple of different tasks. They allow a designer to place a pool of light anywhere on a stage (set permitting), can often change color or the shape of the beam simultaneously, and can provide an accent light that moves with a performer. Before movers, that accent light came from a follow spot, which was human operated. Nowadays, these movers are well integrated into the timecode control scheme that exists in the majority of performance venues, which opens up the same issues with repeatability and perfection with an actor.

The present solutions to this issue are two-fold: hire a human operator to make moves live, or use software. These two solutions are currently visible in large venues, but each is costly; humans are expensive, the software moreso. There are a couple of major software distributors that provide a live-tracking solution: BLACKTRAX, Follow-Me, and ZacTrak are all live solutions with prices beginning at $10,000 and increasing with venue size. Neither solution is cost-effective for the minor production outfits who don't have the space for a follow spot and its operator, or the money to shill out for the nice software. That is where my project aims to fit. I believe that the hardware present across all tiers of production is more than capable of tracking a performer, and by creating open-source software, I aim to provide a solution for those minor production outfits to take advantage of the technology that is available, and to work to try and restore some liveness back into this timecode world.

Chapter 2

Oops all Cameras

2.1 Introduction

Now that we are equipped with the context behind this problem, it's time to dive into the theory and calculations that power my proposed solution. The vast majority of the following content falls under the field of "photogrammetry" or "the art, science, and technology of obtaining reliable information about physical objects and the environment, through processes of recording, measuring, and interpreting images and patterns of electromagnetic radiant energy and other phenomena" i.e. the measurement of life through images. There is a massive open source library for computer vision that will be used to tackle some problems in the following chapter.[1]

2.2 Camera Basics

To start, we must have the same idea on what a camera is, and how we use cameras in this process. The most basic cameras is a pinhole camera, made of a small light-sealed box with a small hole on one end. Light enters the box through that pinhole

1. Kari Pulli et al., "Real-Time Computer Vision with OpenCV," *Communications of the ACM* 55, no. 6 (June 2012): 61–69, ISSN: 0001-0782, 1557-7317, accessed April 4, 2025, https://doi.org/10.1145/2184319.2184337, https://dl.acm.org/doi/10.1145/2184319.2184337.

and creates an image projected on the opposite side of the box.

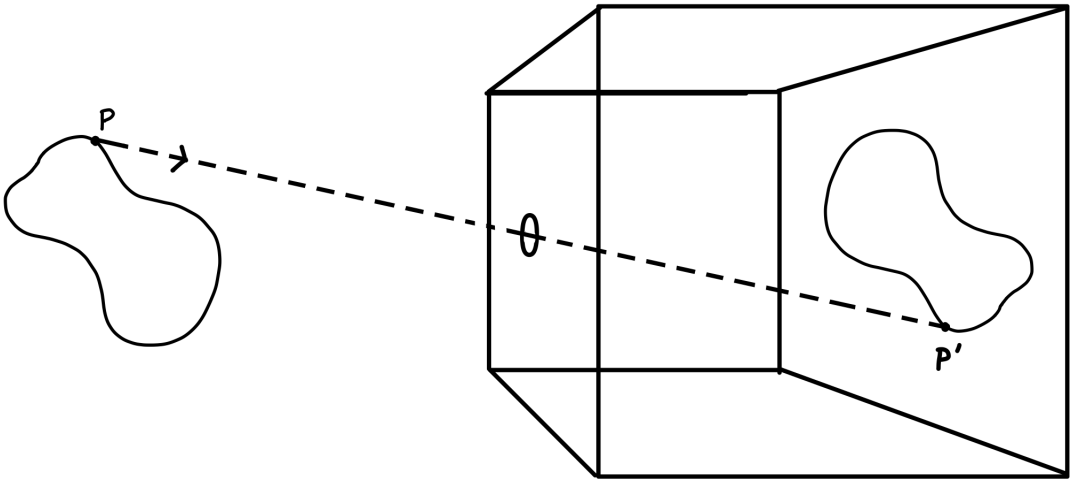

Figure 2.1: A Pinhole Camera

This setup is pictured in Figure 2.1. A couple of things are happening in this image that are being glossed over. For example, the image seen by the pinhole camera ends up flipping the image in the real world. The reason for that is due to how light operates, and why the pinhole camera is effective in recreating the image.

Let's discuss how this image of the world appears as a projection inside the camera. This is depicted in Figure 2.1. Consider an illuminated object sitting in the world in front of the pinhole camera. The light energy hitting the object is diffused, with some rays of light going in the direction of the camera, and others going elsewhere. Figure 2.1 showcases one such ray, reflecting off of point p, which travels through the pinhole, and onto the back face of the camera's body at point p'. Note that in this example, only a single ray is reflected off of point p and makes it through the pinhole, and so p' is where that part of the object is represented.

Tracing similar rays of light off of different points across the object leads to an (upside down) image within the camera. The result of this method is a cone of rays that pass through the pinhole.

Realistically, the pinhole does need to have some width. That results in allowing

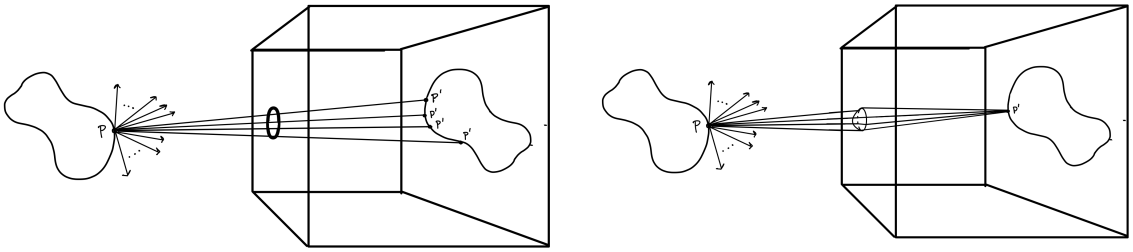

Figure 2.2: Left: A camera with a wide aperture, allowing for light from one point to project in multiple areas. Right: A camera with a lens to refocus the light back to where it should go.

more than a single ray of light to project from a single point. For example, p would have a cone of rays that would project onto the back wall of the camera, which would misrepresent the exact placement of p. However, the smaller the pinhole is, the less light information resulting from each point, thus projecting a dimmer object.

The fix for this is to increase the size of the pinhole, and placing lenses in the created aperture. The intention of the lenses are to focus the cone of light rays from a point p on the object to a single spot p' inside the camera. An example of the lensless and lensed cameras are depicted in Figure **??**.

The cameras that are used today are still based on this pinhole principle, but instead of a backplate, there lies a sensor designed to measure how much light hits a specific zone. Those measurements are shown back to us as pixels on a screen.

2.3 The Camera Model

While this form of pinhole camera may work fine conceptually. We need to define a mathematical model which takes a location of a point in three-dimensional space, and projects it onto a two-dimensional plane. We can represent this process as a matrix:[2]

$$p = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \quad p' = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

2. You may expect the position in n dimensions to be represented with n coordinates. However, we will use homogeneous coordinates which adds an extra component w. Its advantages are forthcoming

$$p' = Mp$$

Where p is a 4×1 vector representing our point in three-dimensional space (or three-space), p' is a 3×1 vector representing the point in two-space, and M is some 3×4 matrix "black box" which will move p to p'. More clarification will come on what that exactly means.

2.4 A (brief) Linear Algebra Primer

Before we get too deep into the world of matrices and various transformations, it's important to establish a baseline understanding of the calculations that are happening behind the scenes. We live in a three-dimensional world–at least that is our perception of the world. Thus, we can represent the locations of things in our world using vectors of three values: (x, y, z). These values represent positions relative to an origin point in distances along three different perpendicular directions. This way of describing a position can be done at home. Take your hand and create the shape of a "finger gun" (where your index finger and thumb are both fully extended), now extend your middle finger halfway (see Figure 2.3 for an example). That creates a coordinate system in terms of your fingers, and you can represent the location of objects in terms of: "how many thumbs, index fingers, and middle fingers does it take to reach an object?"

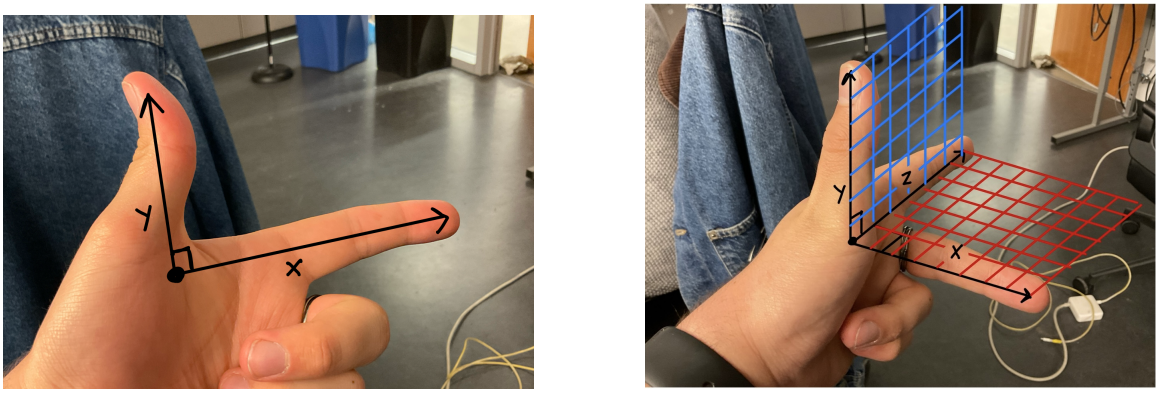

Figure 2.3: Dimensional Fingers showcasing 2-Space and 3-Space

When it comes to the camera, we have two of our directions (the X and Y directions) that are parallel to the X and Y axes of the projection plane and a third direction (Z) which is perpendicular to the projection plane. We can then take locations represented by these vectors, and run them through our intrinsic and extrinsic matrices that define the camera to map the location onto the image.

Notice that the vectors representing the location in three-space and the location on the image plane contain an extra value. That is because these vectors are homogeneous coordinates. Homogeneous coordinates are representations of points in computer graphics (and in our case, photogrammetry) where perspective plays a key role. Remember our representation of p and p' from earlier:

$$p = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \quad \text{and} \quad p' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix}$$

w plays numerous roles, but its primary purpose is the convenience of expressing affine transformations—translation, scaling, and rotation—to be represented by translation, scaling, and rotation matrices.

Translation Matrix as seen in 2.4:

$$q = \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix} \implies q' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 + 1(1) \\ 3 + 1(2) \\ 1 + 0(1) \end{bmatrix} = \begin{bmatrix} 6 \\ 5 \\ 1 \end{bmatrix}$$

Scalar Matrix as seen in 2.5:

$$q = \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix} \implies q' = \begin{bmatrix} \frac{4}{5} & 0 & 0 \\ 0 & \frac{5}{3} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \times \frac{4}{5} \\ 3 \times \frac{5}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ 1 \end{bmatrix}$$

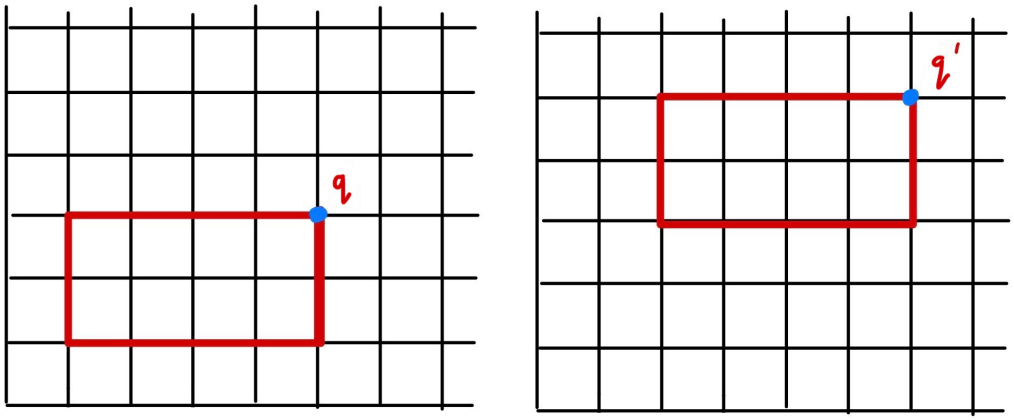

Figure 2.4: Translation of a point q to q'

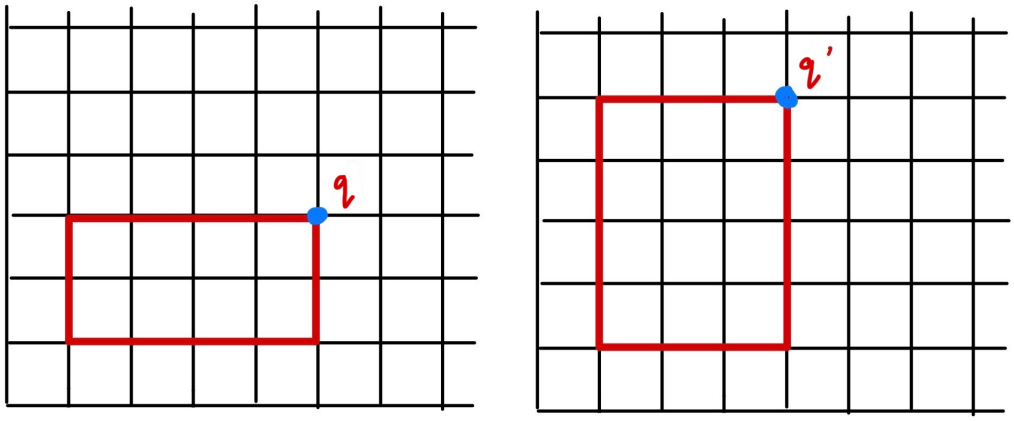

Figure 2.5: Scaling of a point q to q'

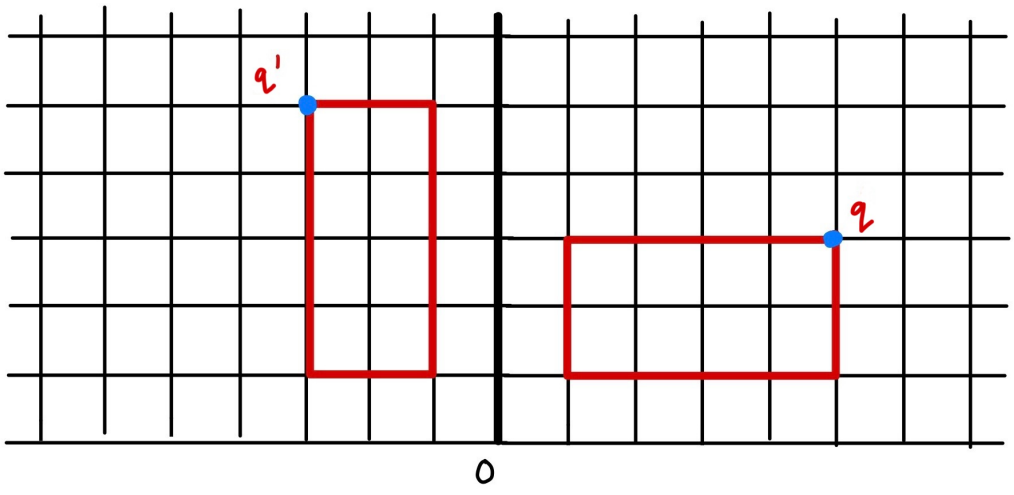

Figure 2.6: Counter-clockwise rotation of a point q to q' by 90°

Rotation Matrix as seen in 2.6:

$$q = \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix} \implies q' = \begin{bmatrix} \cos(90°) & -\sin(90°) & 0 \\ \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5(\cos(90°)) & - & 3(\sin(90°)) & + & 1(0) \\ 5(\sin(90°)) & + & 3(\cos(90°)) & + & 1(0) \\ 5(0) & + & 3(0) & + & 1(1) \end{bmatrix}$$

$$= \begin{bmatrix} 5(0) & - & 3(1) & + & 1(0) \\ 5(1) & + & 3(0) & + & 1(0) \\ 0 & + & 0 & + & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix}$$

2.5 Camera Parameters

We can be more specific about what M is for a camera taking a picture. M is a combination of two matrices that represents different parameters that belong to the camera. Those parameters are classified as camera intrinsics and extrinsics. The intrinsics represent the various distortions and qualities intrinsic to the camera itself. The main values we care about are the focal length, field of view, and optical center, that is, the distance between the lens of the camera and the sensor, the field of view for the camera, and the pixel location for the center of the image. This allows us to take an object in three-space and map it onto pixels in two-space.

We aren't finished yet, because we have no frame of reference to where the camera exists in the world; thus, we need the extrinsics. The extrinsics describe the camera's relation to the world origin. With a single camera, the world origin is usually centered

on the camera itself. The camera extrinsics are represented in rotation and translation matrices. We will cover the calibration part later in this chapter; but the takeaway from this moment is that with these intrinsics and extrinsics that we can calculate through calibration, we can treat the cameras as a "black box" and toss in information about objects in 3D space into the appropriate matrices and the camera will spit out a corresponding image (or collection of pixels representing 3D space). More importantly, we can do this process in reverse, pointing out areas of interest on the cameras' images, and retrieve information on where those areas are represented in 3D space.

The intrinsic and extrinsic parameters of a camera can be represented with matrices. For intrinsic values, they are mapped in the following way:

$$\begin{bmatrix} \text{focal}_x & \text{skew} & \text{center}_x \\ 0 & \text{focal}_y & \text{center}_y \\ 0 & 0 & 1 \end{bmatrix}$$

We call this matrix the "camera matrix" and it is represented as K.

Extrinsic parameters are commonly represented as $[\,R \mid t\,]$, as in, the concatenated matrices of the rotation matrix (which is 3×3) and the translation matrix (which is 1×3). The resulting matrix $[\,R \mid t\,]$ is a 3×4 matrix. With these two matrices, the equation to map a point in three-space to pixel coordinates is as follows:

$$p' = Mp$$

$$p' = K \begin{bmatrix} R & | \, t \end{bmatrix} p$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t1 \\ r_{21} & r_{22} & r_{23} & t2 \\ r_{31} & r_{32} & r_{33} & t3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Cameras are not perfect, however. Lenses contain distortions as light travels through the lens unevenly, especially when going from the center of the lens to the edge of the lens. There are two typical distortions present in our calculations: radial distortion–often dubbed the "fish-eye effect" or, inversely, the "pincushion effect"– and tangential distortion, created by a misalignment of the lens and the image plane. This can be calibrated for, and corrected relatively easily. Lenses have three radial distortion coefficients and two tangential distortion coefficients that can be calculated during calibration and corrected through scaling the x and y coordinates of pixels to compensate.

The final modification we will make to our camera is to establish a "projection plane" for each camera (this will come up again). In short, we create a plane that is a focal length away from the lens in the opposite direction of the camera sensor. Unlike the candle example, where the resulting image is flipped, we are creating a plane for the image to appear "unflipped". This allows us to visualize and explain higher-level concepts more simply, as we don't have to internally flip every image we see.

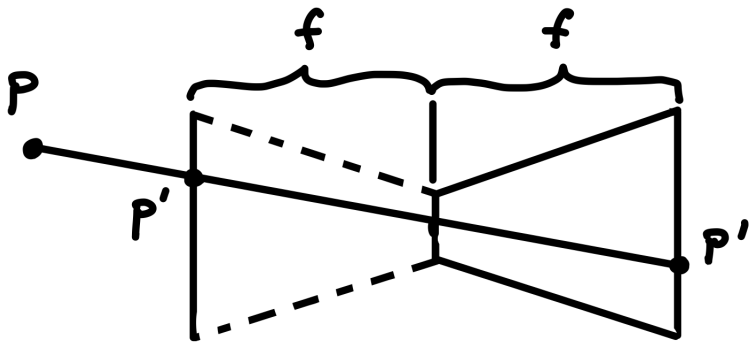

Figure 2.7: Flipping the Projection Plane

2.6 The Stereo Normal Example

Triangulation is relatively straightforward if the cameras are stereo normal (the cameras are adjacent, and on the same plane of X and Z axes). In order to get to DLT, first we should cover the stereo normal base case. Cyrill Stachniss provides a great example, which I will modify for this use case. ([3])

Imagine two cameras c' and c'' which share the same Z and Y axis, pointing in the same direction (specifically the Z axis), with a distance d being the distance between the two cameras on the X-axis. Now, imagine a projection plane that is infinitely large, is perpendicular to the view direction to the cameras, and located a distance f from the cameras on the Z axis. To finish setting up, draw lines z' and z'' perpendicular to the projection plane that go through both cameras respectively. A top-down view of this setup would result in Figure 2.8:

For our stereo normal example, we are going to continue to use this view in our calculations, as we can assume any y or height information in the image will

3. Cyrill Stachniss, dir., "Lecture: Photogrammetry I & II (2021, Uni Bonn, Cyrill Stachniss) - YouTube" (Uni Bonn, 2021), accessed April 4, 2025, https://www.youtube.com/playlist?list=PLgnQpQtFTOGRYjqjdZxTEQPZuFHQa7O7Y.

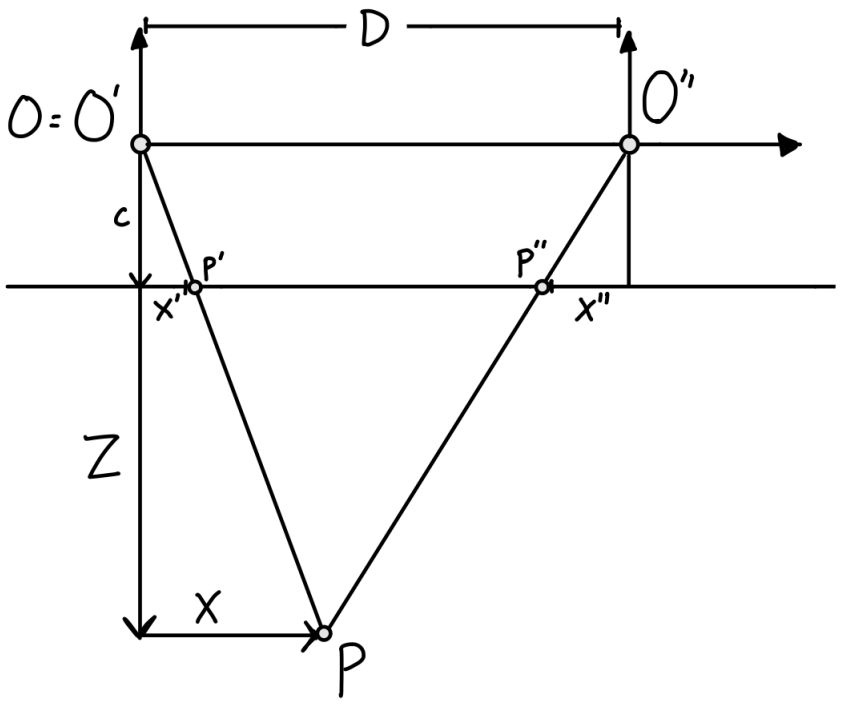

Figure 2.8: A Stereo Normal Example

remain consistent between the two cameras (and in 3D space). This allows for a more simplistic base case, and we can elaborate on how further examples require more thought.

Now, imagine that, on the projection plane, there exists two points that map to where our blob detector has spat out the locations of the LED on each camera. Because we imagine the projection plane to be infinitely large, we map the pixel locations onto this plane using the projection, translation, and rotation matrices we were given in the initial calibration process. We will name these two points p' and p'' which map to cameras c' and c'' respectively. If we draw a line from c' through p' and the same for the other camera, the point at which both cameras intersect will be the location of the LED, which we will call p. To get that result, the math is pretty straightforward.

First, note the distance from p' to z' and p'' to z'' on the projection plane which

we will label x' and x''. Through the intercept theorem, we can find the z coordinate of p by solving the following relation:

$$\frac{z}{c} = \frac{d}{-(x'' - x')}$$

Once that's done, through Law of Similar Triangles, we can find the$x$coordinate of P using z. That relation is as follows:

$$\frac{x}{x'} = \frac{z}{c}$$

Finally, we can assume that the y value of the coordinates is whatever the y pixel value maps to the projection plane. This value is less important for placing the light as we can assume performers stay within the same height for this example.

2.7 Camera Calibration Process

The camera calibration is largely done automatically through OpenCV. The method for calibrating the camera that we use is called Zhang's method.[4] The goal here is to gather the extrinsic and intrinsic parameters of the camera. Remember, the extrinsic and intrinsic parameters of a camera allow us to map an object in three-dimensional space onto a two-dimensional plane. To start, we want to begin with just determining the intrinsic parameters for each camera. We can assume that each camera is in the origin of their own world. The high-level explanation of this process is to choose points whose locations are known in two-dimensional space (the image) and in three-dimensional space (the world). Using that information, and the knowledge that the camera is at the origin, we can reverse engineer the camera matrix (and its

4. Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, no. 11 (November 2000): 1330–1334, ISSN: 01628828, accessed April 30, 2025, https://doi.org/10.1109/34.888718, http://ieeexplore.ieee.org/document/888718/.

parameters). However, this method is easier said than done. The current issues we face is that we want a fast way to calibrate a camera, and we may not have points measured out (in three-space where the camera is the origin). We need to do some tricks.

Zhang's method relies on the usage of a checkerboard with defined parameters; these parameters are the size of squares and the size of the grid (e.g. a 3×5 grid). We assume that the checkerboard is planar–or flat–on some surface. If we are able to detect the corners of the squares on the checkerboard, we could pick a single corner and know where all other corners exist in relation to it. Our first trick is setting the world coordinate system to the bottom-left corner of the checkerboard, with the X and Y axes aligned with the bottom and left sides of the checkerboard and the $-Z$ axis being normal to the checkerboard. Then, for all corner locations on the checkerboard, they all have a single thing in common. The z coordinate of each point in three-space is 0 (or all points lie in the X-Y plane). This trick simplifies the current matrices by a significant amount. Recall that the way to convert a three-dimensional point to a two-dimensional one is as follows:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The board serves as the X-Y plane at $z = 0$, with the Z-axis perpendicular to this plane. Thus, we can simplify the above as follows:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

During this process, we take multiple photos of the checkerboard in view of the camera. For each image we take, we find the bottom left corner of the checkerboard and establish it as the world origin. Then, knowing the size of the checkerboard squares in the real world, we generate a version of the above matrices for each corner in the checkerboard, where K and $[R \mid t]$ are the same. At this point, all we care about is what p and p' are. The OpenCV library takes the checkerboard locations as input. Using its implementation of Zhang's method–which we will treat as a black box for brevity's sake–we can complete our calibration of both cameras.

2.8 DLT to 3D

With the knowledge of the stereo normal example under our belt, alongside the knowledge of how the calibration process works, it's time to create a non-normalized version. With the generation of projection matrices for each camera through calibration, we can theoretically take two points (one on each image), plug their location into the camera matrices, and get a point in three-space that represents the object seen in both cameras. Neither camera can output all the location information about the object on its own. A single camera can calculate the line that would intersect the object, but the second camera is needed to provide feedback on where the object is located along said line.

One method to accomplish this is through a process called Direct Linear Transformation. At its core, it is the process of taking a matrix $Ax = w$ and minimizing w (which allows us to account for noise or imperfect measurements). In this scenario, we know A is the camera matrix containing intrinsic and extrinsic parameters, and we are solving for some point$x$such that w is minimized. Note here that A is representing all the camera matrices, which is done using singular value decomposition. Essentially, the matrices that we described as the properties of the camera are in

special formats that allow them to be merged into a single matrix A. Using SVD and DLT, we can take our two points on the image planes, and find a point in three-space where w (or the distance between the two vectors) is minimized. DLT and SVD are provided within the code, which allows us to treat it also like a black box for now.

2.9 From Pixels to Position

Finally, we are in a position to cover the pipeline that exists in this project. We begin with setting up the cameras in the performance space and calibrating the cameras to the space. The program contains a calibration mode which is modified from Dr. Batpurev's work in stereo calibration. The script takes advantage of the OpenCV library which has in-built functions to calibrate the cameras individually and as a stereo pair through Zhang's method. We take a couple of photos for each calibration step.

Once that calibration is complete, we need to calibrate the cameras to be sensitive to only the LED that we hope to have tracked. Modern representations of images are as a matrix of three vectors, where each pixel corresponds to a set of red, green, and blue (RGB) values (sidenote, OpenCV defaults to a format of green, red, blue (GRB) values). The webcams that are used in this project operate at a resolution of 1080 pixels by 720 pixels, so to analyze a single frame of video data, the computer needs to work with 777,600 three vectors of GRB data. In order to aid the software, we want to filter out the data we don't care about.

I used a blue LED light because it was visually distinct. That way, I could filter out pixels that don't appear as shades of blue. This technique is called "masking", and is pretty straightforward: First, we need to discuss a different representation of color referred to as HSV. While BGR (and other representations of RGB color) store the image as representations of the amount of red, green, and blue appear in a pixel,

HSV approaches color through hue, saturation, and value. Hue represents the color, and is a value from 0-359. Colors that feature red range in the hue values of 0-60, green in the range of 60-180, blue in the range from 180-300, and back to red in the range of 300-359. The peak or most pure forms of red, green, and blue are at hues 0, 120, and 240, and allows for easy translation between RGB and HSV. Saturation represents the distance from white a color sits, and is a range from 0-100. Value is the distance from black a color sits.

HSV is actually great for image processing because it interprets color in a similar way the brain perceives color. For example, we do not see a color as a specific mix of red, green, and blue light, and when comparing two shades of the same, or similar color, we may say that one is darker or lighter, which HSV primarily encodes. In this use case, if we want to mask the image down to a certain color, we can first convert all the pixels to an HSV format, then go pixel-by-pixel, and look for pixels that fall within a certain range of hue. Because the LED light reflects off of objects, we can also filter by saturation and values to remove darker blues and focus purely on the brighter shades coming directly out of the LED.

Figure 2.9: Left: Image of LED, Right: Mask of LED by HSV

When we have the masked frames, the next step is to run a blob detection algorithm. The basic idea is to search the mask for blobs. A high-level description of the algorithm is to search the mask for pixels of similar color. Because the mask

results in an image with only fully black or white pixels based on the hue of the LED, the light emanating from the LED results in a large clump of pixels. The blob detection algorithm allows the user to filter out blobs based on certain qualities (like the roundness of a certain blob). In our case, because we are putting so much work into filtering the image before running it through the blob detection, we allow the detector to be lenient and to flag any blobs. The detector outputs a list of all blobs it detects, specifically the image coordinates of the center of each blob. With those image coordinates, we can input into our DLT solver that we declared before. Because we calibrated the cameras, the vast majority of the work is done already. We know the camera matrix and the rotation-translation matrix for each camera, so we can input the pixel location on each camera and get a point in three-space representing the location of the LED.

Now that we have this location, the final step is to pick a location for the instrument and to transmit it to said instrument. Theoretically this is pretty easy. If we know the instrument's position in relation to the world we have defined with the cameras, we can convert the location of the LED to polar coordinates. At this point, there are defined values that exist for the lighting instrument to use, and all that is left is to transmit it. I take advantage of the numerous existing python libraries that exist that allow someone to transmit a universe of data through a serial port. One of the most common serial ports that people encounter is the USB port (which stands for Universal Serial Bus). We can plug in an off-the-shelf USB to DMX decoder which—through the magic of some electronics cleverness—will output a DMX signal that runs on a five-pin DMX cable to the instrument. Rinse and repeat at least fifteen times a second and we have a functional performer tracking program!

Chapter 3

Bringing the Project to Life (?)

The original idea was to demo the tracking software in Belle's thesis, Chronicle of a Planet. The show explores the various relationships between Belle and mother. Those representations were her own mother, her motherland as a student from China, and her mother tongue of Mandarin.

The original goal was to include the tracker in the chapter exploring motherland. In that chapter, Belle moves around a white umbrella. The intended effect was to track the umbrella in certain sections when Belle was criticizing the CCP most, to serve as the "watchful eye of a surveillance state". The show was set to open on February 14th.

3.1 Where we started (spring semester):

I had ended the semester feeling a bit disheartened. At that moment I was hoping for a fully functional demo of the tracking software ready to plug into a moving light. I realize now it was a lofty task. Over that winter break, I spent some time digging around the internet to see if someone had done work with stereo vision to see what I was missing; this was when I discovered the work by Temuge Batpurev, a researcher with a PhD in Computer Science from Osaka University. Dr. Batpurev

had spent time working with stereo vision, and had developed a script to calibrate stereo cameras. The issue with my project so far was that even if I had gotten a good track with the stereo cameras, the data coming out with my back-of-the-napkin math made no sense for the current context. That was because of two things: I hadn't calibrated for the distortion that a camera experiences through lenses and the fisheye effect, and I tried to rely on perfect placement of the two cameras with a measured distance instead of calibrating the two. Dr. Batpurev's script handled both of those problems.

I began the semester integrating and testing Dr. Batpurev's code. His code relies on calibrating through using a checkerboard with a specific size and row/column count. In fact, opencv provides infrastructure for detecting a checkerboard for this exact purpose (!!!). The integration proved to be fairly straightforward. The next challenge was to send instructions over DMX.

3.2 DMX "Hell"

DMX is an interesting protocol. The protocol itself is a serial protocol, which means that bytes are transmitted sequentially, one bit at a time. Serial protocols are quite common in the world of communications; one of the best known ports that exist on a computer is the USB port or Univeral Serial Bus port. USB is quite versitle, and allows for basically anything to be transmitted across it. There are speed limitations, but for a protocol as basic as DMX, this is a non-issue While I had done some work in networking & protocols, my knowledge of implementing it in python wasn't the greatest. Thankfully, through the magic of open-source software, there were multiple people who had created python modules designed to work in DMX transmission. One of the defining characteristics of DMX was that every packet of data to be transmitted *had* to be 513 bytes; the first one is the **start code** and it denotes the way the

signal should be interpreted (usually 0x00). The other 512 bytes communicate the behavior of the following 512 addresses, with one byte assigned to each address. Because of this, the majority of open-source software I encountered would have me either create the entire 513 byte packet, or to define what addresses were needed for the light (and what data it used).

The light for this was a ROBE Esprite, provided by Outlaw Lighting–A local supplier of lighting gear. It required, on average, 49 addresses to run, to control various aspects of the light. That made for an interesting evening of starting at datasheets provided by ROBE, and using the magic of object orientation to create a consistent way to *serialize* all the attributes I needed. I ended up getting it working on its own, but I wasn't let off that easily. One of the main issues is dealing with traffic. An unforseen challenge was managing the data traffic from two webcams providing input, and a third port providing DMX output.

3.3 So how did you get so close?

I felt really good going into the final week before tech (the period where all technical aspects of a show are combined). I had a way to communicate with the light, and I had a way to track the LED. Sure, I needed to still find a way to communicate the position of the LED to the Esprite, but even if I could just start to feed junk data, and just see what happens, it would have been awesome. Wishful thinking.

In order to try and speed up the tracking to be as close to real-time as possible, I downgraded the resolution from 1920x1080 to 640x480. That's a difference of 2,457,600 pixels to analyze to 307,200 pixels (all at a hopeful speed of 20-30fps). The tracking works great when close up, but I was hoping for Belle to be dancing with the umbrella more than 15 feet from the camera, and with that resolution, the cameras had a *bit* of trouble even putting the pinprick of light that was the LED on screen.

Oops.

However, if I tried to upscale the resolution, it would scale the calculations by a factor of at least 8. I am not the most efficient programmer, and some of the algorithms I used absolutely did not run in constant time (maybe elaborate on wtf that means). Even when I tried that, the computer struggled to keep pace (and also the LED did appear at a far distance very well).

Unfortunately, at that point, tech arrived and I resolved to put the project on hold while I finished designing for Belle's thesis.

3.4 Where does that put us now:

Well, it was certainly a step back, but the encouraging thing is that there is a pipeline that exists. After discussing with my advisors, the best step forward was to try to work on the pipeline in sections to figure out the capacity, and to generate demos for each particular step. That way, even if the final, full pipeline never came to completion, I would still be able to showcase my work in the various sections. The three demos Jim and I came up with were as follows:

1. [ ] A working demo showing my ability to write programmed sequences for a moving light to follow. This showcases my ability to just write onto DMX from my laptop.

2. [ ] A working demo tracking an LED with stereo camera, and displaying all the points (or points found every second) on a graph. Likely using matplotlib

3. [ ] A working demo that takes a generated graph (like one created through the previous demo), and sends it to the light.

If I can get all three working, then the pipeline should be trivial. This still means I need to work through calibrating the light, but I have some ideas on how to accomplish that. Although we are facing obstacles, this is a two step forward, one

step back process.

Conclusions & Reflections

As of writing, this project stands incomplete. I was able to get some mapping of a 3D point onto a graph, but the current issue stands with the moving light. During Belle's thesis process, I was able to experiment with a ROBE Esprite, an industry standard moving light with the pan and tilt directly moving the luminare. As we only had the budget to use the Esprite for the production, the current option is a Rosco ICue, which is an attachment that clips onto an ETC Source 4 Elipsoidal. The ICue is a moving mirror, which uses a two-axis gimbal to move a mirror to reflect/redirect the beam of light. That introduces some complications into the calibration of the light in preparation for a technical demo. That is the immediate step I plan to take in this process.

Had I the ability to start this process over, I would recommend starting with Dr. Stachniss's Photogrammetry course, which would have allowed me to spend more time digging around under the hood of the black boxes used to calibrate cameras, and locate objects.

The idea is still a sound one, and absolutely accomplishable in the remaining time in the semester. It is just a manner of getting into the space and doing the work.

There are many places I hope to take this software upon it's completion. For starters, the current implementation has communication with a lighting instrument in isolation from the rest of the lighting network. While this is not an issue for my specific project, it means that the instrument cannot be controlled by a conventional

lighting console. One of my original dreams for this process was to create an injector/extractor duo which took two DMX lines as input: one from the console, one from my software; the injector would interweave the signals together, encoding the signal with a different start code so all other lights would ignore its instructions. The extractor would sit between the DMX chain and the moving light, and listen for instructions destined for the instrument with either start codes. Once recieving those instructions, the extractor would reencode the start code for the instrument, and forward the instruction. This is also possible using network switches and a dedicated conversion into sACN, where the instrument could be referred to by IP address.

Regardless of the result for this project, it will be released publicly with an open source license. I have benefitted much from the availability of code from my peers and the least I can do is pay it forward. This stereo vision also has more applications beyond just lighting tracking. With calibrated cameras, one could do all manner of performance art. Thus, take a swing at my software. Tweak and modify and make it your own.

Appendix A

The Code of the Project

To be added...

All code can be accessed at https://github.com/gabeah/thesis

Appendix B

Photos of the demo (?)

To be added...

Bibliography

Ahrens, Lynn, and Robert Viagas. *The Alchemy of Theatre : The Divine Science : Essays on Theatre & the Art of Collaboration*. In collaboration with Internet Archive. New York : Playbill Books : Applause Theatre? ; Milwaukee, WI : Distribition, North America, Hal Leonard Corp., 2006. ISBN: 978-1-55783-698-4, accessed October 4, 2024. http://archive.org/details/alchemyoftheatre0000ahre.

Albertz, Joerg. "A Look Back; 140 Years of Photogrammetry." https://www.asprs.org/wp-content/uploads/pers/2007journal/may/lookback.pdf.

Birringer, Johannes. "Dance and Media Technologies." *PAJ: A Journal of Performance and Art* 24, no. 1 (January 2002): 84–93. ISSN: 1520-281X, 1537-9477, accessed September 13, 2024. https://doi.org/10.1162/152028101753401811. https://direct.mit.edu/pajj/article/24/1%20(70)/84-93/55164.

Bowman, Wayne, and Jean Bowman. *Modern Theatre Lighting*. New York: Harper, 1957.

Cadena, Richard. *Automated Lighting: The Art and Science of Moving Light in Theatre, Live Performance, Broadcast, and Entertainment*. Amsterdam ; Boston: Focal Press, 2006. ISBN: 978-0-240-80703-4.

Culjak, Ivan, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. "A Brief Introduction to OpenCV." In *2012 Proceedings of the 35th International Convention MIPRO*, 1725–1730. 2012 Proceedings of the 35th International Convention MIPRO. May 2012. Accessed April 4, 2025. https://ieeexplore.ieee.org/document/6240859.

Dixon, Steve. *Digital Performance : A History of New Media in Theater, Dance, Performance Art, and Installation*. Leonardo. Cambridge, Massachusetts ; The MIT Press, 2007. ISBN: 0-262-04235-5.

Dupré, Sven. "Introduction the Hockney-Falco Thesis: Constraints and Opportunities." *Early Science and Medicine* 10, no. 2 (2005): 125–136. ISSN: 1383-7427, accessed April 4, 2025. JSTOR: 4130307. https://www.jstor.org/stable/4130307.

Frenzel, Louis E. *Handbook of Serial Communications Interfaces : A Comprehensive Compendium of Serial Digital Input/Output (I/o) Standards*. Chantilly, UNITED STATES: Elsevier Science & Technology, 2015. ISBN: 978-0-12-800671-9. http://ebookcentral.proquest.com/lib/reed/detail.action?docID=2189944.

Guarnieri, Massimo. "Switching the Light: From Chemical to Electrical [Historical]." *IEEE Industrial Electronics Magazine* 9, no. 3 (September 2015): 44–47. ISSN: 1932-4529, accessed October 4, 2024. https://doi.org/10.1109/MIE.2015.2454038. http://ieeexplore.ieee.org/document/7271159/.

Hartley, Richard., and Andrew. Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge, UK ; Cambridge University Press, 2003. ISBN: 0-521-54051-8.

Huntington, John. *Introduction to Show Control: Connecting Entertainment Control Systems for Live Shows*. NYC, USA: Zircon Designs Press, 2023. ISBN: 978-1-7357638-4-2.

Internet Archive, collab. *The Renaissance Stage : Documents of Serlio, Sabbattini and Furttenbach*. Coral Gables, Fla. : University of Miami Press, 1958. ISBN: 978-0-87024-004-1, accessed December 9, 2024. http://archive.org/details/renaissancestage0000unse.

Keller, Max., Johannes. Weiss, Michael. Robinson, and Andreas. Klotz. *Light Fantastic : The Art and Design of Stage Lighting*. 2nd rev. and updated ed. Light. Munich ; Prestel, 2006. ISBN: 978-3-7913-3685-5.

"Lighting." In *The Companion to Theatre and Performance*. Oxford University Press, January 1, 2010. ISBN: 978-0-19-957419-3, accessed April 29, 2025. https://www.oxfordreference.com/display/10.1093/acref/9780199574193.001.0001/acref-9780199574193-e-2307.

Opencv/Opencv. OpenCV, April 4, 2025. Accessed April 4, 2025. https://github.com/opencv/opencv.

"OSC Spec 1_0," April 7, 2021. Accessed April 29, 2025. https://opensoundcontrol.stanford.edu/spec-1_0.html#introduction.

Pilbrow, Richard. *Stage Lighting Design : The Art, the Craft, the Life*. New York: By Design Press, 1997. ISBN: 0-89676-139-8.

Pulli, Kari, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. "Real-Time Computer Vision with OpenCV." *Communications of the ACM* 55, no. 6 (June 2012): 61–69. ISSN: 0001-0782, 1557-7317, accessed April 4, 2025. https://doi.org/10.1145/2184319.2184337. https://dl.acm.org/doi/10.1145/2184319.2184337.

Randall Wright, dir. "BBC, David Hockney's Secret Knowledge." November 21, 2021. Accessed April 4, 2025. https://www.youtube.com/watch?v=R-0UXBcjlRY.

"RP-002-014_v1-1-1_MIDI_Show_Control_Specification_96-1-4.Pdf." Google Docs. Accessed April 29, 2025. https://drive.google.com/file/d/1Hbix4KbuQRysnEF6L3sXhlwJtc4RE33L/view?usp=drive_link&usp=embed_facebook.

Serlio, Sebastiano. *Five Bookes of Architecture: Translated out of Italian into Dutch and out of Dutch into English*. In collaboration with Glasgow School of Art Library. Robert Peake, 1611. Accessed December 9, 2024. http://archive.org/details/firstbookeofarch00serl.

Seul, Michael., Lawrence. O'Gorman, and Michael J. Sammon. *Practical Algorithms for Image Analysis : Description, Examples, and Code*. Cambridge ; Cambridge University Press, 2000. ISBN: 0-521-66065-3.

Simpson, Robert S. *Lighting Control: Technology and Applications*. Taylor & Francis, 2003. ISBN: 978-0-240-51566-3. Google Books: gS1EfPUOTUoC.

Stachniss, Cyrill, dir. "Lecture: Photogrammetry I & II (2021, Uni Bonn, Cyrill Stachniss) - YouTube." Uni Bonn, 2021. Accessed April 4, 2025. https://www.youtube.com/playlist?list=PLgnQpQtFTOGRYjqjdZxTEQPZuFHQa7O7Y.

Swann, Alex. "Sound Design : A Systems Approach," Reed College, 2016.

Tannahill, Jordan. "Why Live?: A Question for 21st Century Theatre." *World Literature Today* 90, no. 1 (2016): 36–39. ISSN: 0196-3570, accessed April 24, 2025. https://doi.org/10.7588/worllitetoda.90.1.0036. JSTOR: 10.7588/worllitetoda.90.1.0036. https://www.jstor.org/stable/10.7588/worllitetoda.90.1.0036.

"Theatrecrafts - Equipment - Grand Master Board." Accessed April 29, 2025. https://www.theatrecrafts.com/bhc/equipment/strand-grand-master-board.

"Theatrecrafts - Equipment - Grand Master Board." Accessed April 29, 2025. https://www.theatrecrafts.com/bhc/equipment/strand-grand-master-board.

Wild, Larry. *A Brief Outline of the History of Stage Lighting*. Northern State Univerity, 2015. https://www.terryewell.com/m114/Readings/BriefHistoryStageLighting.pdf.

Zhang, Z. "A Flexible New Technique for Camera Calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, no. 11 (November 2000): 1330–1334. ISSN: 01628828, accessed April 30, 2025. https://doi.org/10.1109/34.888718. http://ieeexplore.ieee.org/document/888718/.