Contents

Laser Stabilization Math Writeup

Hyrum Taylor

August 18, 2025

1 Introduction

When stabilizing a laser beam, our goal is to take in four measurements from the two cameras (X-axis and Y-axis beam deviations for both cameras), and use these measurements to determine how much to move the four motors. This is a difficult problem because moving one motor impacts both cameras: for example, moving X-axis motor 1 by 10 steps might move the beam in the X direction by 1 pixel on camera 1, and 2 pixels on camera 2.

This document is meant to provide a mathematical description of this process, as well as providing a physical interpretation of the math. While understanding the math is not strictly necessary for implementing the code on a new system, understanding it will help develop system setups that will respond better.

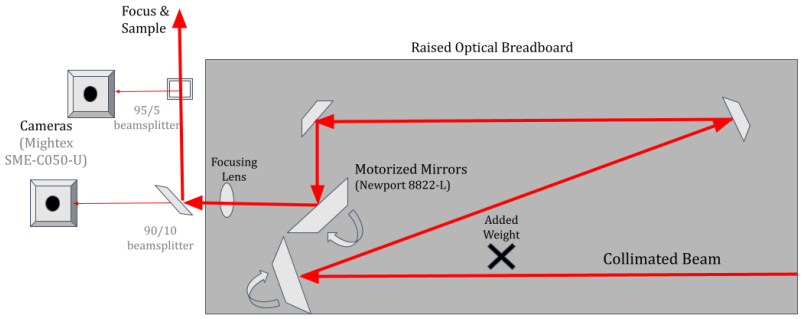

Figure 1: Stabilized Setup Diagram: Diagram of the test HeNe setup with two motorized mirrors stabilizing the beam on two cameras. The two cameras (on the left) record where the beam is, which is then used to decide how much to move the two motorized mirrors (with the arrows to demonstrate movement) in order to correct for beam drift. Lengths and angles in this diagram are only approximate, but the raised optical table is 30 cm by 90 cm. The weight used to introduce the error is placed at the black X. This image is taken from *H. Taylor, "Methods for Improving tabletop Ptychography", Bachelor's Thesis, BYU, 2025.*

2 Simplifying Assumptions

The first assumption that we make is that a movement by an X-axis motor has no impact on the beam's Y-axis, and vice versa. While this may not be the case, the movement in one axis due to movement of a motor for the other axis will be small, meaning that we can neglect it with very little loss of accuracy.

The second assumption that we make is that motor steps are the same size in both directions. This is, again, not necessarily true (my measurements show up to a 10% difference in step size going forward vs. backwards), but the mistakes that this causes are small compared to other sources of beam deviation. When taking calibration measurements (the numbers that tell the ratio of motor steps to pixel movement for a given camera-motor pair), I recommend taking the average calibration number for forward and backward movement.

Without these assumptions, we would be working with an excessively large number of calibration values, for very little to no improvement in performance. We would require 32 calibration values, instead of the 8 that are currently used.

3 Conversion Matrix

If it weren't for the assumption above, we would need a 4x8 matrix (X and Y deviation on two cameras, and four motors * forward vs. backward). Due to the assumptions we made, we can instead have two 2x2 matrices: one for the X-axis, and one for the Y-axis. For the rest of this document, we will only work on the X-axis, but the math works equally well for the Y-axis.

To begin with, let's define the conversion matrix M so that:

$$\begin{aligned} \begin{pmatrix} camera_1 \\ camera_2 \end{pmatrix} &= M \begin{pmatrix} motor_1 \\ motor_2 \end{pmatrix} \\ M = \begin{pmatrix} m_1 \to c_1 & m_2 \to c_1 \\ m_1 \to c_2 & m_2 \to c_2 \end{pmatrix} &= \begin{pmatrix} m_1 c_1 & m_2 c_1 \\ m_1 c_2 & m_2 c_2 \end{pmatrix} \end{aligned} \tag{1}$$

In other words, we're letting M be a matrix that converts two motor step lengths into two camera pixel shifts. Thus, it has units of pixels/motor step.

It is fairly easy to create M: This is the purpose of *auto-calibration.py*. When the stabilization code is running, we need the inverse of M in order to turn measured shift on the camera into the number of motor steps.

$$\begin{aligned} M^{-1} &= \frac{1}{det(M)} \begin{pmatrix} m_2 c_2 & -m_2 c_1 \\ -m_1 c_2 & m_1 c_1 \end{pmatrix} \\ \begin{pmatrix} motor_1 \\ motor_2 \end{pmatrix} &= M^{-1} \begin{pmatrix} camera_1 \\ camera_2 \end{pmatrix} \end{aligned} \tag{2}$$

Notice that this matrix has units of motor steps/pixel, as we would expect. We can now use this matrix in real time to take the beam deviation in one axis

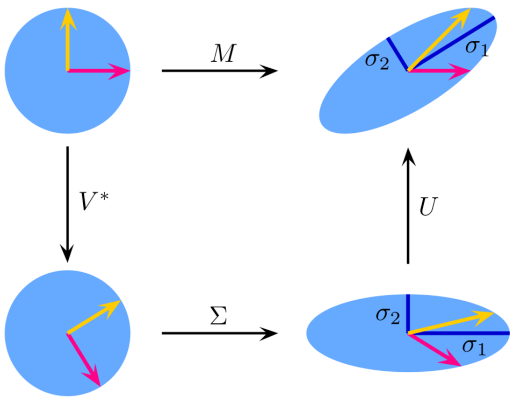

Figure 2: Diagram showing visually the role of each matrix in the SVD. Notice that σ determines the length of the major and minor axes of the resulting oval. Taken from Wikipedia, "Singular Value Decomposition", en.wikipedia.org/wiki/Singular_value_decomposition, accessed Aug. 18, 2025

on both cameras, and use this to find how much to move the motors by in order to reset the beam location.

4 SVD Analysis

This section is a quick refresher on Singular Value Decomposition (SVD). If the reader is already familiar with SVD, feel free to skip to the next section.

SVD is a mathematical process to turn one matrix into a product of three matrices. As we are working with square matrices, all three matrices are the same shape as the original (2x2). Performing SVD on M gives us the equation

$$M = U\Sigma V^* \tag{3}$$

where U and V are unitary matrices, and Σ is a diagonal matrix. As M has only real values, all three resulting matrices are real as well. The values along the diagonal of Σ are non-negative, and are traditionally ordered from largest to smallest. Also, remember that any unitary matrix A has the property that $A^*A = AA^* = \mathbb{I}$.

Finally, let's remember that for a unitary matrix with real values, $|det(A)| = 1$, and that for a diagonal matrix D with diagonal values $d_0 ... d_n$, we have $det(D) = \Pi_{i=0}^{n} d_i$. Then we know that $|det(M)|$ is the product of the singular values.

5 When Will Stabilization Fail?

Due to the fact that we need to use M^{-1} to convert camera pixel movement into motor steps, we need M^{-1} to exist. If $det(M) = 0$, then M^{-1} is infinity, meaning it cannot be used. As we discussed in Section 4, $|det(M)| = \Pi_{i=0}^{n}\sigma_i$, where σ_i is a singular value of M. Then, we must not allow any of the singular values to equal zero, or the stabilization code will not work.

Before talking about what this means physically, let's introduce another way of looking at this problem. Let's describe the X-axis as a 2D vector space; where motor movements, camera shifts, and beam position/angle are three valid ways of creating a basis for the space. Note that this is why we use two motorized mirrors and two cameras - we cannot cover the vector space with one of each, and three of each is unnecessary. Then, our goal is to keep the beam at the origin of this vector space. The matrix M transforms us from the motor basis to the camera basis, and M^{-1} transforms us back.

Look back at Figure 2, and in particular notice that the singular values are associated with the major and minor axes of the new vector basis. In other words, there is a fourth vector basis that is directly associated with the singular values.

Now, let's start putting everything together. This fourth basis is the natural one to talk about whether or not the code is able to stabilize the beam. Here are a number of insights that we can draw from looking at the 2D X-axis vector space from this basis.

- If a singular value is zero, then the vector space that the motors are able to move the beam to has only one dimension, not two.
  - It makes sense that M is not invertible in this case: In all but a very few cases, beam deviation cannot be corrected for by the motors because the deviation is outside of the space that the motors can reach.
- The larger the small singular value, and the closer the two values are together, the easier it will be for the code to place the beam back at the center of the vector space
  - I noticed this experimentally: when the small SVD value is large enough, the PI controller took less time settling down to the origin.
  - This can be used as a useful comparison tool between different camera or motor placements. In general, the larger the second singular value, the better the stabilization process will be.
- The (orthogonal) vectors that form the fourth basis associated with the singular values can be found in the columns of the U and V matrices.
  - I believe that the vectors found in the U matrix are associated with the camera basis, and the vectors in V are associated with the motor basis, although I haven't verified this.

6 Applications to an Optical Setup

6.1 Camera Placement

If both cameras are placed at the same point in the beam, (for example if both are placed at the focus), a few mathematically equivalent things will happen:

- The cameras will record identical beam movement. This means that the information will be duplicated between both cameras. Essentially, we are only measuring one degree of information, instead of the required two.
- We will find that $m_1c_1 = m_1c_2$, and $m_2c_1 = m_2c_2$.
  - Then $det(M) = 0$
- The vector space for both the X and Y axes (both of which are 2D) will not be able to be measured by the cameras. Even though the motors can move the beam into whatever position it wants, the cameras will not be able to fully characterize the beam.
- The code will be unable to stabilize the beam.

For this reason, I placed my cameras so that one is at the focus of the beam and the other is slightly before. The more separation, the better, so long as the beam can still be accuratly recorded. We need to be careful of both oversaturation and undersaturation, as both can destroy the fidelity of the collected data. Also, the beam must stay fully on the camera at all times.

6.2 Motorized Mirror Placement

If the motors are placed very close together, then movement on one motor will have negligibly different impacts from movement of the other mirror. If so, then these things will happen:

- Moving one mirror will have the no different impact from moving the other mirror. Essentially, the beam only has one degree of freedom per axis, instead of the required two.
- We will find that $m_1c_1 = m_2c_1$, and $m_1c_2 = m_2c_2$.
  - Then $det(M) = 0$
- The vector space for both the X and Y axes (both of which are 2D) will not be able to be covered by the motors. Even though the cameras will be able to tell where the beam has deviated, the motors will be unable to move the beam back to the original location.
- The code will be unable to stabilize the beam.

For this reason, I placed one motor close to the focusing lens, and the other far upstream. I have not yet had problems with motor steps being too large to correct for beam deviations that the camera can see, but if $m_i c_j$ gets too small (likely in the range of 2-3, although that may change depending on the setup), it may be necessary to move distant motors closer to the cameras.

7 Conclusion

Thinking about the stabilization code as a process that attempts to bring the beam back to the origin in two vector spaces simultaneously (X and Y vector spaces) is a useful way to think about the process. This treatment provides a natural way to bring motor movement, camera measurements, and beam position/angle into one coherent whole, while also providing a way to numerically compare the ability of different setups to stabilize a beam.

If there are any questions, feel free to contact me at hct10000@gmail.com.