# A Mechanical Analogy to Adaptive Optics: The Automatic Ping-pong Bouncer

Xiaotian Zhang (Jim), mentored by Dr. Humphrey Wong

August 2015 to May 2016

**Abstract**

In our independent study, we studied the process by which modern ground telescopes adjust for atmospheric interference when observing the night sky (a process called adaptive optics) and constructed a mechanical contraption that performs the analogous task of dropping a ping pong ball from a random height which is a task complete with an error signal, adjustment for the error signal using mathematical modelling, and execution of the mathematical adjustment.

## 1 Inspiration from Adaptive Optics (AO)

By reading papers and looking at telescope operator websites, we found that there were 4 steps to create a clearer telescope image:

1. Measuring the initial image
2. Measuring the error signal by laser
3. Modelling the wavefront of the image using a Zernike polynomial
4. Adjusting for this error by using MEMS actuators and deformed mirrors.

Before we constructed an analogous structure with affordable materials, we first looked into the actual product of an AO telescope - this proved to be too costly. Then, we looked at an inspiring contraption from ETH Zurich that balanced a pencil on its tip - an impossible task for humans. They utilized 2 orthogonal motion-detecting cameras for about USD$2000 each. We then thought of the idea to bounce a ping-pong ball into a basket, no matter how far or how high the ping pong ball was dropped. The tools required were relatively inexpensive and readily available:

- Arduino Uno board, breadboard, and wires ($5 from Alibaba's Taobao, shipping included)
- Sonar sensor ($10 from Amazon, shipping included)
- A Hi-Technic servo used in FTC Robotics
- 80/20 Aluminium extrusion
- Karate-chopped thick wooden board
- Duct tape
- 3-star ping pong ball

## 2 Construction of the Contraption

We attached 3 orthogonal bars of equal-length aluminium extrusion at their endpoints using several L-brackets designed specifically for this purpose. This was the base and also where the servo would be mounted. We attached the servo to the top of the extrusion bar and allowed it to rotate in the x-z plane (see figure).

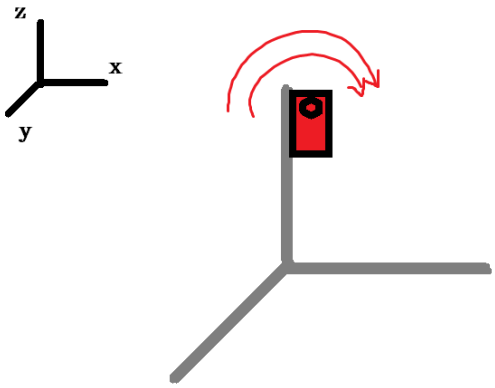


Figure 1: A diagram of our aluminium extrusions and mounted servo.

We then attached a thick wooden board to this servo so that the board could rotate from being flat in the x-y plane to being flat in the y-z plane (see figures).

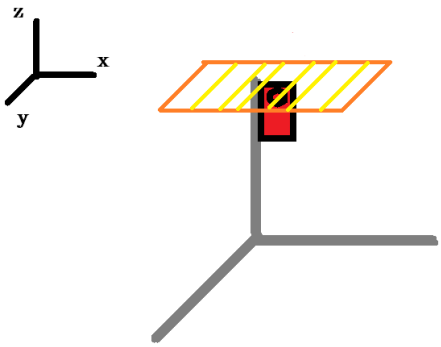


Figure 2: A diagram of our contraption with emphasis on the board's rotation.

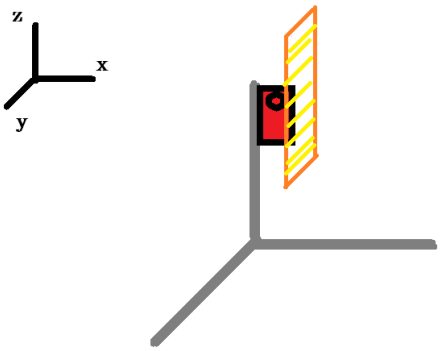


Figure 3: A diagram of our contraption with emphasis on the board's rotation.

The board can be attached to the servo in any way, as long as it is sturdy, so we chose to use aluminium extrusion with duct tape.

We attached the sonar sensor at the end of our aluminium extrusion bar in the y-axis and pointed it upwards to measure the height at which a ping pong ball was dropped.

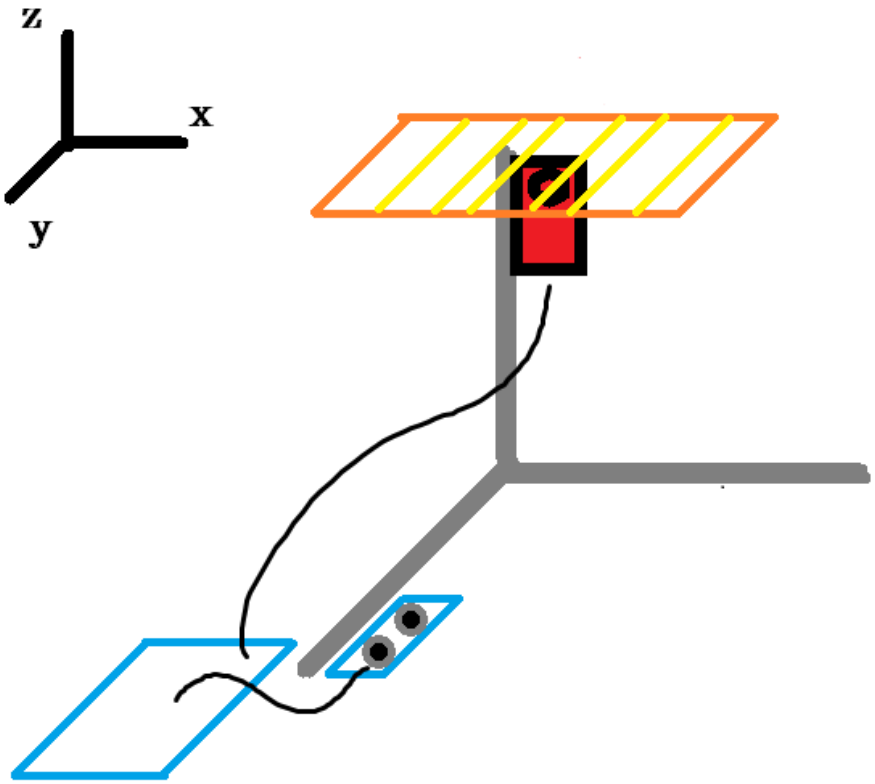


Figure 4: A diagram of our completed contraption with sonar sensor, Arduino board, and wiring in place.

## 3 Measuring the Initial Data

To measure the initial data, we dropped ping pong balls from different heights and different distances to the box and measured the angle that the servo required to perform this task properly. Since it is easier to modify the drop height than to modify the angle, various trials were made to find the correct drop height for a set angle. For every distance from the board to the target, at least 5 height-angle pairs were recorded, with all minimum heights occurring at 40 degrees of tilt. Note that this is not the actual angle with respect to the x-y plane, but rather a reading given by the servo. In reality, it is a 24 degree tilt, as measured by the iPhone 5's gyroscope.

To summarize: data comes in the form of distance to the target, the height at which ping pong balls were dropped, and the recorded angle of the servo rotation that was needed to get the ball in the box. We found that for every height, there were usually 2 possible angles that could get the ball in the box. One of these occurred between the range of 0 to 40 degrees, and the other between 40 and 90 degrees. This distinction becomes very important later on in the correction phase since it causes the correction function to not be a function.

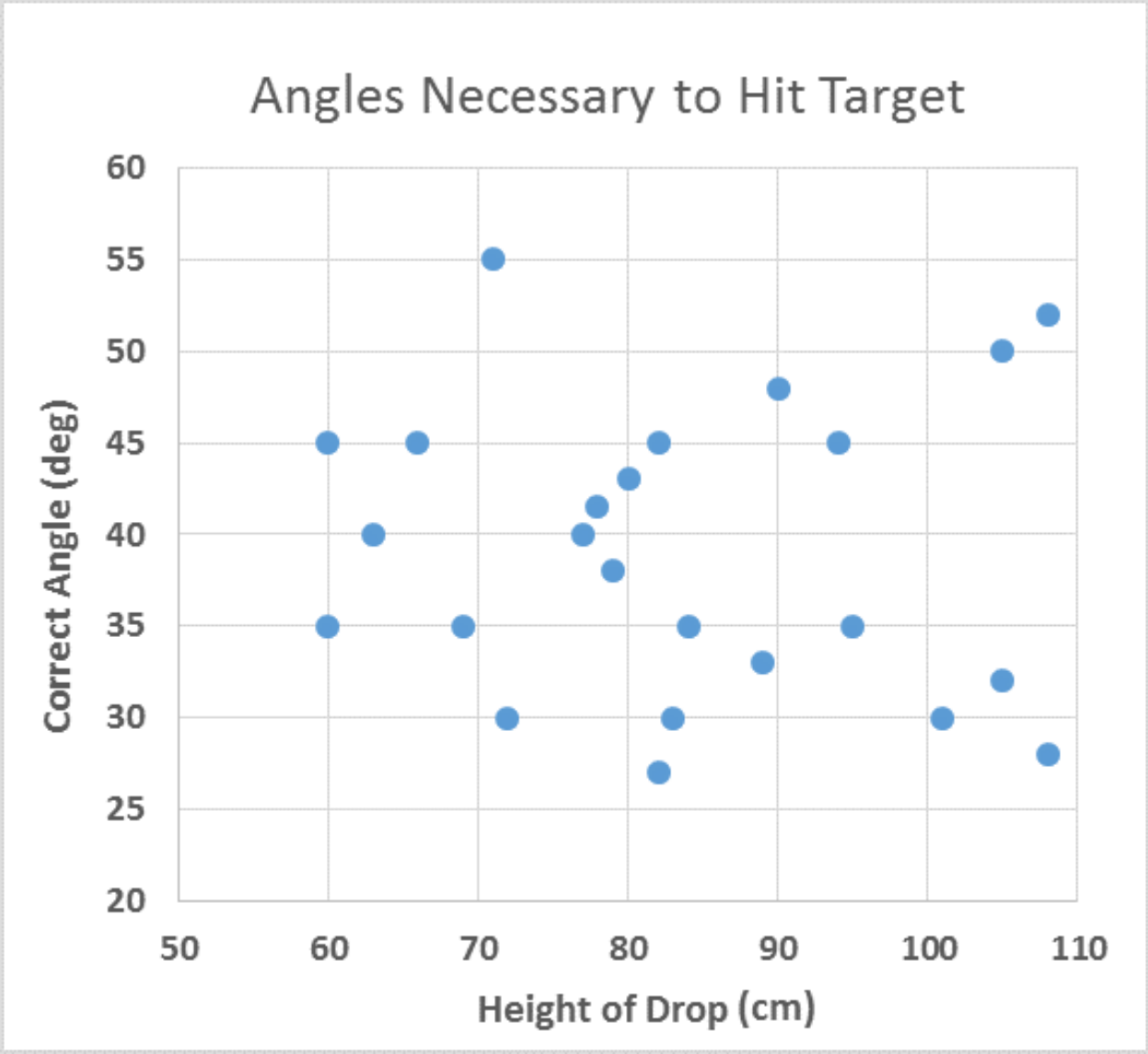


Figure 5: A plot of the angle required to bounce the ping pong ball into the target box at various heights and distances to the box.

# 4 Measuring and Modelling the Error

The error in the case of the ping pong reflector is the correct angle(s) needed to reflect the ping pong ball to its target correctly, minus the current angle of the board. The angle necessary to do so is shown in the plot above, but since there are two possible angles for every height, the data needs to be split at the midpoint of the possibilities (where a certain height yields only one valid angle). By visual inspection, this midpoint seems to be at 40 degrees. We can therefore split the data into a set of data for angles between 0 and 40 degrees, and a set of data for angles greater than 40 degrees.

## 4.1 The Equation

In AO, Zernike polynomials are used to model the wavefront of the iamge. For our contraption, we can model each set of data using some function with inputs of drop height and distance, and output of the necessary angle. The most suitable equation for this purpose is the projectile range equation simplified for projectiles

fired on level ground, which is given by:

$$d = \frac{v^2}{g} \sin 2\theta, \tag{1}$$

where $d$ is the distance to the target, $v$ is the initial velocity, $g$ is the gravitational acceleration, and $\theta$ is the initial angle of launch. When we have a ping pong ball dropped from some height $H$, the conservation of energy equation $\frac{1}{2}mv^2 = mgH$ suggests that $v^2 = 2gH$. So, if we perform a substitution in equation 1, we have:

$$d = 2H \sin 2\theta \tag{2}$$

Solving for $\theta$ and adding some constants to get a good fit later on, we have:

$$\theta = C_1 \arcsin\left(C_2 \frac{d}{H} + C_3\right) + C_4 \tag{3}$$

We can add or remove constants as we like and test their fits, but as an example, we can include all four and solve for the best constants.

## 4.2 Optimizing the Constants

One can use Matlab to solve for the best Zernike polynomial coefficients to fit the wavefront. Existing Matlab code uses the method of least squares to find these coefficients. For our contraption, we use a numerical method of least squares and MS Excel's Solver add-in to find our own constants $C_n$. The Solver add-in is free, but usually requires the user to activate it inside the preferences.

First, we set up the data columns for distance, drop height, and correct angle in Excel. To the immediate right we create a column called model, and to the right of that a column called error (actually, squared error). We have our constants to the right and reference them with double dollar signs. We define the model according to equation 3, the error as the model column minus angle column, squared. The total error is the sum of the error column. We minimize the total error box by changing the values for C1 to C4, allowing negative values. We try this method for both sets of data (0-40 degrees and 40+ degrees) and see which set has the least error relative to the number of data entries (0-40 degrees wins).

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | **distance** | **height** | **angle** | **model** | **error^2** | | **vars** | |
| 2 | 36 | 101 | 30 | 29.99814 | 3.48E-06 | | C1 | 0.654556243 |
| 3 | 36 | 84 | 35 | 35.56383 | 0.317906 | | C2 | 0.940341301 |
| 4 | 36 | 79 | 38 | 38.24246 | 0.058785 | | C3 | 0.517678748 |
| 5 | 36 | 77 | 40 | 39.60497 | 0.156051 | | C4 | -0.145010056 |
| 6 | 36 | 108 | 28 | 28.48713 | 0.237295 | | **7.262305** | <-TOTAL ERROR |
| 7 | 36 | 89 | 33 | 33.51887 | 0.269225 | | | |
| 8 | 30 | 63 | 40 | 40.71603 | 0.512694 | | | |
| 9 | 30 | 69 | 35 | 36.13532 | 1.288958 | | | |
| 10 | 30 | 83 | 30 | 30.33905 | 0.114955 | | | |
| 11 | 25 | 72 | 30 | 29.38405 | 0.379389 | | | |
| 12 | 25 | 60 | 35 | 34.52219 | 0.228304 | | | |
| 13 | 25 | 82 | 27 | 26.74255 | 0.06628 | | | |
| 14 | 40 | 88 | 40 | 38.11766 | 3.543202 | | | |
| 15 | 40 | 95 | 35 | 34.89838 | 0.010327 | | | |
| 16 | 40 | 105 | 32 | 31.71905 | 0.07893 | | | |

Figure 6: A screenshot of the data table used to create the model for angles less than 40 degrees.

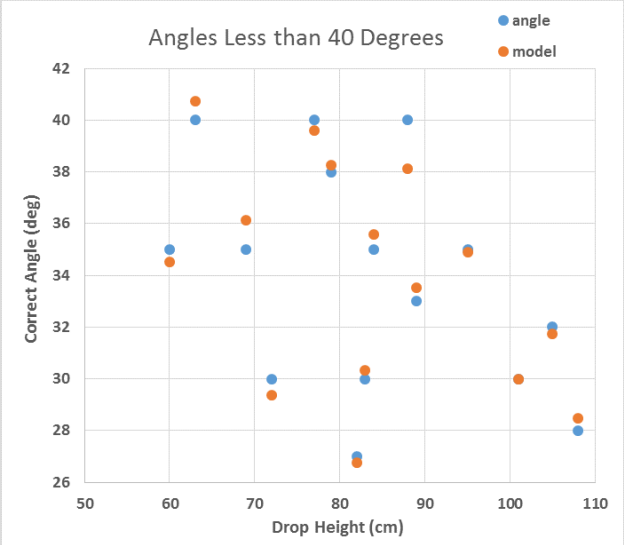


Figure 7: A plot of our data and model for angles less than 40 degrees.

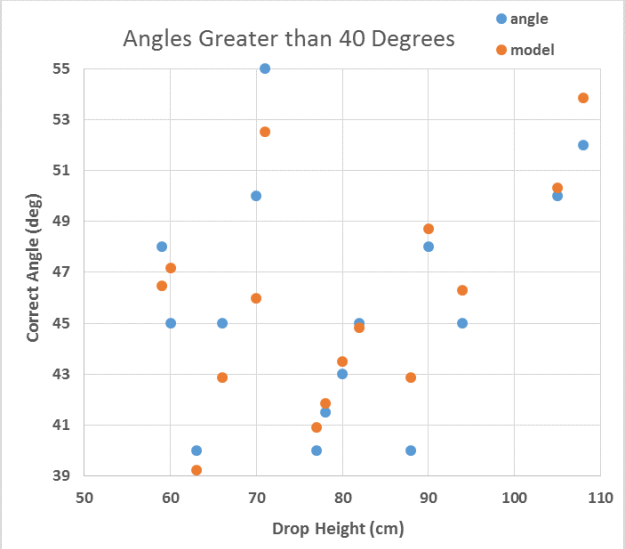


Figure 8: A plot of our data and model for angles more than 40 degrees. The error is visibly greater for each data point.

## 5 Adjusting for Error

In an AO telescope, the many MEMS actuators are used to deform the telescope mirror in order to fix the atmospheric effects on the image. This is done using the Zernike polynomial used to model the wavefront. For us, we use the servo to tilt our wooden board in order to correctly bounce the ball into the box. This is done using the range equation with solved coefficients.

Now, instead of making the servo rotate to some fixed angle, we have an equation to make it always bounce the ball into the bucket correctly. This equation is equation 3, with constants $C_n$ being determined by our least-squares fit in the earlier step. We can actively measure the drop height using our sonar sensor pointed upwards to the hand, and adjust our servo angle according to the equation. One downside of using only one sonar sensor is that the distance to the box has to be manually entered into the system. The Arduino code relating to the equation looks something like this (for angles less than 40 degrees):

```
double angle = 180/PI*(0.654556243*asin(0.940341301*distance/height+0.517678748)-0.145010056);

if(angle>0&&angle<41){
   S.write(angle);
   delay(500);
}
```

Delay makes sure the servo doesn't twitch madly, and the if statement filters out invalid drop heights.

## 6 Conclusion

We have constructed a contraption capable of bouncing a ping pong ball into a bucket regardless of height or distance (unless it is out of range). This was done for relatively low costs compared to other adaptive technologies such as a pencil balancer, a ball balancer, or a complete AO telescope. Our Arduino code and video can be found bundled together with our paper.