# Computer Vision: Sentry Nerf Gun

Carlo Colizzi, Gabby Blake, Ben Grant

October 2022

Figure 1: Assembled Nerf Sentry system

# 1 Introduction

The goal for this project was to use computer vision to create a Nerf sentry gun. A sentry gun is a weapon that is automatically aimed and fired at targets that are detected by sensors.[1] We also wanted to mount this Nerf gun on a Neato robot vacuum, so that our system could move around various locations autonomously. Moreover, we decided to create two modes for our project: one that identifies, tracks, and aims at a specific color, and one that uses facial recognition to do the same.

# 2 System Design

## 2.1 Mechanical

We opted to use a Nerf Stryfe[2] as the gun component in our design, as it propels darts with electrically powered, opposed flywheels. This meant that we would only need to actuate a servo motor to pull the gun's trigger and a relay to close the flywheel circuit, which was a significantly easier option than designing for a spring-powered gun requiring a lot of cocking force.

We decided to mount the Nerf gun on a custom-made pan/tilt turret with both axes driven by NEMA-17 stepper motors. With this design, the sentry gun would be able to hit targets in a semi-spherical targeting envelope such as the one pictured to the right in figure 2. We decided to 3D print this structure, as this method gave us a high degree of flexibility and efficiency in our developing stages.

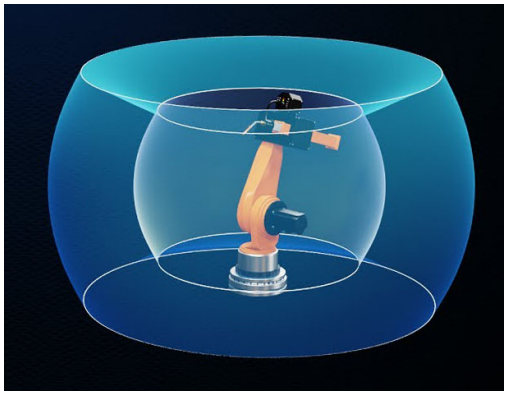


Figure 2: Work envelope of a typical robotic arm

Our most complex 3D-printed part was the crossed-roller bearing used to support and drive the pan-axis of the pan/tilt turret (figure 3). This type of bearing can support both axial (thrust), radial, and combined loads while remaining free-spinning. In case we needed to install the pan/tilt stepper motors on the turret asymmetrically, we needed a low-cost, 3D-printed bearing solution that could handle combined radial/axial loads without binding.

We also set a design goal of making as few permanent modifications to the Nerf gun as possible, so we designed our components to fit onto the existing attachment rails of the Stryfe.

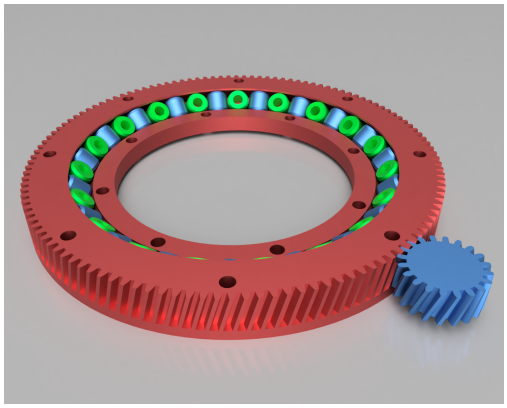


Figure 3: 3D printable crossed-roller bearing

[1] Wikipedia
[2] Hasbro

## 2.2 Electrical

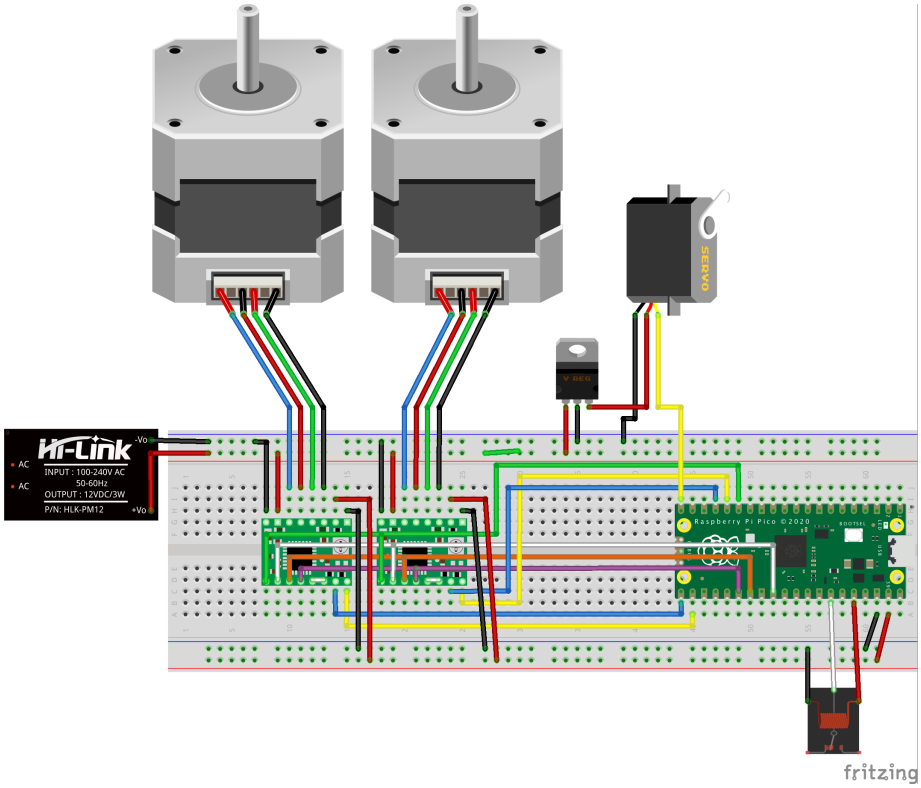


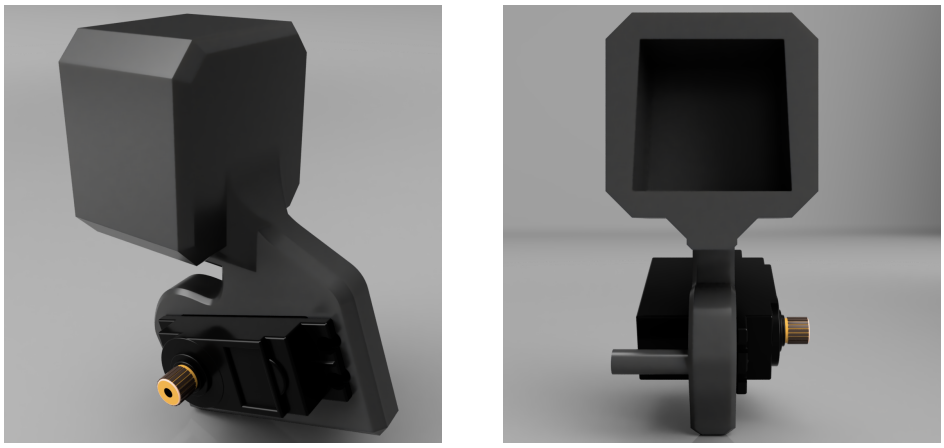

Figure 4: Wiring diagram

The actuators we used in the Nerf Sentry system were:

- Two NEMA-17 stepper motors for driving both axes of the pan/tilt mount
- A 5v relay for controlling the Nerf gun's flywheel motors
- A servo motor for pulling the Nerf gun's trigger

Our goal for the electrical design of the project was to interface to have the entire Nerf Sentry system able to be controlled by means of a single Micro-USB serial connection. This goal necessitated the use of a microcontroller for controlling two A4988 stepper motor drivers, the relay, and the servo. We made use of a Raspberry Pi Pico running CircuitPython code to listen for serial commands and drive the actuators accordingly.

## 2.3 CAD and Fabrication



(a) Side view (b) Front view

Figure 5: Trigger puller servo mount

Firstly, we 3D printed an attachment for the back of the gun which fits snugly onto a protrusion on the back of the Nerf gun (figure 5). This served as an attached housing for a servo motor to pull the Nerf gun's trigger. On one side of the housing, we have an anchor point for one end of a string, which is then passed in front of the trigger and tied to a gear arm on the servo. This way, whenever the servo turned, the string would be pulled and the trigger engaged.

After mounting this first attachment we used the rails on the Nerf gun to mount a custom attachment for a PiCam (figure 6).

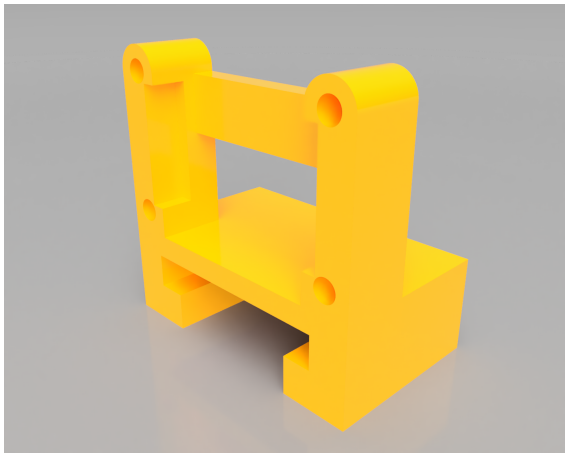


Figure 6: Picam rail-mount

Lastly, we made the actual pan/tilt structure. Again, we decided to use the rails on the gun to our advantage, using them as a mounting for the attachments. This was made so that the gun would rotate around its approximate center of mass, as to minimize stress on the motors running the pan/tilt interface. Furthermore, we also cut a hole into the middle of the print, so that the gun could freely spin and move without the cables getting in the way or getting tangled.

Figure 7: Pan/tilt assembly

# 3 CV Targeting

## 3.1 Object Identification

We created two different object identification functions: one that uses color masking to identify a particular color, and one that uses face detection.

### 3.1.1 Face Detection

The face detection function, *find_faces()*, uses OpenCV to convert the input to a binary image and an XML file to identify facial components in it. Some rectangles are then drawn around each identified face. During this stage, we encountered an unexpected behaviour: the script was identifying many smaller "faces" in the image background, even when they weren't there. To account for this, we decided to create a lower limit to how big the rectangles have to be for their content to be considered a face. Once we have identified the faces in the image, we take the center of the rectangle and consider that the centroid of the face. These coordinates are then passed down to the *process_centroid()* function.

### 3.1.2 Color Masking

The color masking function, *find_shapes()*, works quite similarly to the face detection one. Again, we use OpenCV to convert to a binary image and identify the objects that match the color requirements, in our

case red objects. We then find and return the centroid of these objects and pass the coordinates to the *process_centroid()* function.

### 3.2 Object Tracking

The *process_centroid* function receives the centroid (x- and y- coordinates) of the target as input and evaluates it. If the centroid is within the turret's line of fire, represented by the centroid being within the bounds a small blue box fixed at the center of the camera frame, then velocity commands are sent to the pan and tilt motors of the turret to aim it at its target. These velocity commands are determined through PD controllers and sent over serial (further explained in the next section *Thinking and Acting*). If after receiving the target centroid coordinates, it is determined that the target is already within the aiming box, the command to fire is sent over serial to the motor that controls the trigger of the turret.

### 3.3 Thinking and Acting

Once we have a set of coordinates for the centroid of the object we're tracking, we used a PD controller for both the pan and tilt motors respectively to move the camera until our target was at the center of the image. A PID controller is a proportional-integral-derivative control loop mechanism that is used to minimize the error between an measured process variable and desired setpoint. The proportional constant determines the amplitude of oscillation that occurs when the process variable approaches the setpoint value, goes past it, and then changes its values to go towards that setpoint value again before repeating the process over and over until a minimum error is reached. This process can sped up though by changing the integral and derivative constants to dampen this oscillation quickly and effectively. We did not change the integral constant, so it is more accurate to say we implemented a PD controller. The derivative constant was used to dampen the oscillation caused by the P constant. In the end the values of the constants were $K\_p$ is -0.001 and $K\_d$ is 0.0001 and the limits of the controllers were -1 to 1 m/s as we did not want the velocities of the motors to be any faster than this 1 m/s in either direction. The turret's camera was physically located above the barrel of the gun so while the camera was locking in on faces, the turret would fire further down at the chest of the target individual.

## 4 Discussion

When working on this project we have run into various unexpected problems. One of these was the fact that whenever the script detects more than one face, the centroid will be at the average location of these. In order to correct this, we decided to add some code to use only the first identified face as the centroid.

One other issue that we ran into was serial communication with the Pi and the Pi Pico and getting that communication to work through ROS. We were able to make these mechanisms work independently, for example the facial recognition, color masking and controlling the motors through keyboard commands all worked, but we struggled to connect the pieces. After many hours of trying to do this, we decided to cut out the middle man and scrap the Neato and the ROS architecture, putting all the code into one module and running one program. With this decision, we were able to integrate the turret and computer vision code, and the turret was able to move according to what it saw through the camera.

# 5 Future Improvements

Although this project turned out great, there is still lots of room for improvements. If more time was available, we would have integrated our sentry Nerf with the Neato robots, allowing our machine to move around a location autonomously. This was in our initial plan but due to unexpected issues with running our code on a Raspberry Pi and using ROS at the same time we decided to develop our project without this capability. Another improvement that we would focus on is the facial recognition. Although it does work as intended, the algorithm still occasionally identifies faces where there aren't any. Having more time to work on this would allow us to tweak our code, resulting in a more accurate face detection. Lastly, if we had more time we would also work on integrating our code to run without the need of a laptop. Although the Pi is technically able to run the code, it is very slow and therefore unusable for our project. Possibly including multi-threading and overclocking could allow us to optimise performance on the Pi.

# 6 Conclusion

We believe one of the key takeaways from this project were the use of `ansycio` library to run multiple steppers at the same time: We found that this python library greatly helped us achieve a smooth and efficient movement. It helped us run PWM signals to the stepper motors simultaneously, so that the gun moved in a diagonal rather than moving solely in the x and y dimensions.

Another takeaway was learning so much about integrating software, mechanical, and electrical components of a project and just how difficult it can be. Although with different equipment we probably would have had a much easier time, it also important to be mindful of scoping a project realistically. I do feel however that by shooting for a deliverable that we knew would be very difficult, we accomplished and learned so much more than if we were cautious and picked a simpler idea for the sake of easiness, which we are very proud of.