Sentence Writing Assistant Network: SWAN

Jasper Eng and Firas Al Chalabi

Abstract

This project aimed to create an efficient system for user interaction with a system capable of writing legible sentences. The goal was to design a highly precise robot using cost-effective resources, such as Lego Ev3 motors, and wood. SWAN's design was inspired by 3D printers, and underwent iterative refinement for small, precise movements. The software components included path planning for shaping letters and implementing different font sizes, along with a computer vision system using Convolutional Neural Network architectures for letter detection from images. The project successfully achieved its primary objective of having SWAN write sentences and integrated a word detection program, though room for improvement in accuracy and portability is acknowledged for future work.

1. Motivation

The goal of this project was to design a system that allowed a user to interface with a robot that would be able to write clean legible sentences. The inspiration for this project was the fact that our own handwriting is quite bad, so we thought it would be interesting to see if we could create a system that was able to outperform our own writing. We also enjoyed the idea of creating highly precise (mm precision) robots with cheap and accessible resources.

2. Hardware

This section will provide a comprehensive list of parts that we used, and also how they were used in our design.

2.1 Components

We used: 1 Lego Ev3 Intelligent Brick, 3 Lego Ev3 Large motors, two counterweights one weighing 3lbs and the other 5lbs, 1m of fishing line, 3 pieces of hardwood flooring, one two by four, one drawer pull unit, and lastly a sliding tray. In addition to these parts we used a few Lego pieces to supplement the build.

We used the motors as well as our counterweights and fishing line in order to get movement of our end effector in the x, y, and z direction. All of the wood in the build was used to design the frame on which the end effector was mounted. The sliding tray and drawer pulling unit were both used for fixing movement in one direction. Since both of these moved in fixed straight lines they were ideal for ensuring that when we rotated the motors we ended up with straight lines.

3. Design

The design of our robot, SWAN, turned out to be critical to achieving the degree of accuracy that we needed. We went through many design ideas before deciding on our final build, this included a robot arm, a Cartesian robot with a smaller robot mounted at the end effector for precise movement and finally our design of a Cartesian robot with 2 prismatic joints. We took inspiration from 3D printer design, since they are also built for small precise movements.

In order to turn the rotational motion of the motors into the linear motion requiired, we spooled fishing line into coils above each of our motors. This, of course, would only allow the motor to pull an object towards itself, so in order to get movement in the opposite direction we used a counterweight attached to the other end of the string. This meant that when the motor unspooled the fishing line, the counterweight would pull the object in the other direction. This design also came with the benefit that it was highly precise, allowing us to use the rather inaccurate Lego motors and still achieve millimeter precision. We believe the reason that the cables work so well is that they do not suffer from problems such as gear lashing.

With the ability to use our motors for linear motion we were able to create movement in the x, and y direction of our page. To do this we fixed our fishing line to objects that were fixed in their movement i.e. our sliding tray, and drawer slider. The sliding tray was used to hold our paper and served as our writing surface, while the drawer slider was used to mount our end effector and the additional motor we used to get movement in the z direction. We then mounted the drawer slider to a wooden frame letting it sit over top of our writing surface.

The last step in our build process was to get z move-

ment so we could lift our pen up from the page. To do this we fit the pen into a tube mounted to our drawer slider, then we inserted a pin into the top of our pen. Since the distance we needed to travel was quite small (¡ 3cm) we could simply attach a bar across the Lego motor and to the pin in the pen, then when the motor would rotate the pen would move up and down. With that completed we now had a robot that was accurate enough to perform the task outlined above, and we could begin work on designing the software.

The design outlined above is a description of our final build, each of the design aspects laid out went through many iterations of trial and error. Some more than others, for example one design aspect that caused us the most trouble was getting the pen to lift up – the z motion. The first iteration of this design relied on gears, but those proved to not be robust enough to even minor shifts in our environment. After many different design ideas we finally settled on the pin design highlighted above.

4. Software

There were 2 main software components in this project: path planning and computer vision.

4.1 Path Planning

4.1.1 Drawing Simple Shapes

To be able to draw any particular letter we needed to be able to draw the shapes that comprise that letter. We noticed that all uppercase letters in the English alphabet can be broken down into 4 main shapes: horizontal, vertical and diagonal lines, as well as quarter ellipses. Note that diagonal lines can face forwards or backwards, and that quarter ellipses can have any orientation based on the 4 quadrants of a Cartesian plane (see Figure 4.1 below).

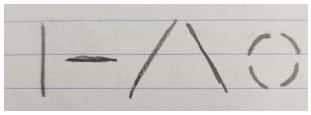

Figure 4.1: The simple shapes that comprise all uppercase letters.

The way we draw horizontal lines is by moving the servo motor attached to the drawer slider (subsequently referred to as the x-motor), and vertical lines by moving the servo motor attached to the sliding tray (subsequently referred to as the y-motor). Assume that if we want to draw a straight line that is 1 unit long in a specific direction we rotate the corresponding motor by θ°. The direction of rotation determined the path that would be taken when drawing a particular shape. We drew diagonal lines by moving both motors at the same time, and we made sure that both motors stopped at the same time by adjusting the ratio of their speeds according to how long we wanted the diagonal line to be in both the x-direction and the y-direction. For example, if we wanted a diagonal line that extended 0.5 units in the x-direction and 2 units in the y-direction, we would rotate x-motor and the y-motor 0.5θ and 2θ respectively, while simultaneously moving the x-motor at a quarter of the y-motor's speed (as $0.5/2 = 1/4$). As for quarter ellipses, we broke them down into a series of diagonal lines with different lengths in the x- and y-directions. We calculated those lengths programmatically using the sine and cosine functions. Drawing accurate quarter ellipses was very challenging as a lot of error was accumulated when making series of small movements with the inaccurate Ev3 motors.

4.1.2 Drawing Letters

Once we were able to draw the simple shapes that comprise the uppercase letters (see above), the next step was to break down each letter into those shapes (see Figure 4.2). We decided that each letter was to be bounded in a box that was 2 units long and 1 unit wide (see Figure 4.3), and that we can assume that the pen will always start at the bottom left corner of said box. This way, we can draw each letter without having to factor in our current position on the paper, or account for the previous letter drawn.

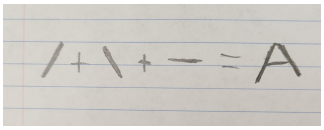

Figure 4.2: Breaking down the letter A into its components.

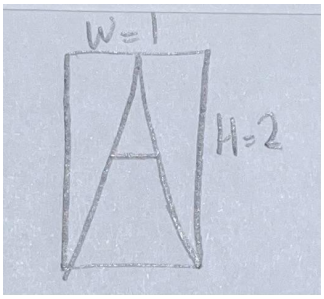

Figure 4.3: An example of the letter A in its bounding box.

4.1.3 Writing Words and Sentences The way we designed our letter-drawing workflow made it relatively easy to write complete words. Since, for each letter, the pen would start and end on the bottom left corner of the bounding box (in the pen-up position), moving on to the next letter was simply a matter of moving in the x-direction by 1.25 units. We determined that 0.25 units was an appropriate spacing between a letter and the next by testing different values. To get spaces between words, we just "skip" a letter, i.e. we lift the pen up and move the motor the width of one letter from where the next letter was supposed to start. We keep track of the number of letters we have drawn per line, and when we're about to exceed that amount, or we encounter a newline character, we move our pen down (in the y-direction) 2.25 units, and we move back the width of the number of letters including the spaces between letters.

4.1.4 Implementing Different Font Sizes We wanted SWAN to write words and sentences in different font sizes. We started by defining our maximum font size, which maps 1 unit in either the x- or y-directions to 1 full rotation of the corresponding motor. Since our font size is an integer between 1 and 5 (inclusive), then for a font size x, we rotate our motors $(360 * x/5)^\circ$ to move 1 unit in either direction. The dimensions of our workspace were 14 cm (width) by 25 cm (height). We measured the length of a line drawn by 1 full rotation of our x- and y- motor, 3 times, and found an average length of 1.9 ± 0.05 cm, whereas for our y-motor, one full rotation corresponded to a line of length 1.7 ± 0.05 cm. The reason for this difference is probably due to the difference in spooling of the fishing wire between the 2 motors. This means that we can draw $\lfloor 14/(1.9 * (x/5) * 1.25) \rfloor$ letters per line at font size x. Similarly, the numbers of lines per page can be calculated using the expression $\lfloor 25/(1.7 * (x/5) * 2.25) \rfloor$.

4.2 Detecting Letters From Images

Once we were able to draw letters, we wanted to create multiple different interfaces for the user to interact with SWAN. The most basic way to do that is to pass a string for the robot to write, through a terminal. To further improve user experience we also incorporated an image recognition interface. This is a system where the user can input an image of a word they would like the robot to write.

4.2.1 Pre-Processing The first step in this process is to get characters from an image. To do this our system reads the image in grey scale, and then performs a colour thresholding operation in order to identify all parts of the image that are letters. We can then draw bounding boxes around each of the regions of interest. The issue with doing this on the original image was that humans do not write each letter evenly spaced, meaning that the bounding box for one letter would often contain parts of another letter, see fig 4.3.

Figure 4.4: This image shows how parts of other letters can be present in the bounding boxes of different letters. In the upper left corner there is an artifact from a different letter in the bounding box of the "a".

To solve this problem we project the letters whose region of interest (ROI) we were looking at onto a blank black image. This removes any extra noise in the image, and only leaves the outlined letter. The last step in the pre-processing step is to make sure we have the correct ordering of letters. To do this we put the bounded images into a list sorted by the x coordinate of their upper left corner. Meaning that we now have a list of images that goes from the leftmost letter to rightmost.

We can now take this list and pass each image into a classification model in order to determine which letter we are looking at. Before doing that we make sure that all images are the correct size and format for our model to accept.

4.2.2 Classification The next step in our computer vision pipeline is to classify an image of a letter. To do this we tried two different classification architectures, a Visual Transformer (ViT), and a Convolutional Neural Network (CNN). The ViT broke the image into 4 patches, which were then passed through a linear layer before being passed to the transformer. The transformer consisted of six residual self attention blocks. The output of the transformer was then passed to a classification network, which was just a linear layer with an output size of 47 - the number of classes in our data set. The CNN was comprised of 3 convolutional layers of decreasing size, followed by 2 fully connected layers, and lastly an output layer again with an output size of 47.

4.2.3 Training Both of these models were trained for 50 epochs on the EMNIST data set, which contains images of handwritten letters and numbers. The EMNIST data set has a variety of different "splits" which allow you to group the data differently. We choose to use the balanced split, this – as the name implies – ensures that we have equal samples from each of the 47 classes. The classes provided in this split are all of the numbers, upper case letters, and most of the lower case letters, some lower case letters are re-labeled as upper case letters since they are difficult or impossible to distinguish from each other, eg. lower case "s" and uppercase "S". After training we tested both models on some of our own handwritten images, and decided that the CNN model provided greater real world accuracy.

5. Results

At the end of the project, we were able to achieve our primary objective: writing sentences with our robot SWAN. We also achieved a secondary goal: integrating a written word detection program. The assembly of the robot was the most challenging part, as we implemented a unique design that had not been previously implemented (at least not to our knowledge). There definitely is a lot of room for improvement, such as increasing the accuracy of SWAN, and making it more compact and portable. However, with the time and resources we had, we are happy with what we came up with and what we achieved.

6. Future Work

In the future, we would most probably work on editing the design of SWAN to improve accuracy and portability.
There are many ways we can improve accuracy. One such way is to use more accurate motors. We can also use a different mechanism to convert rotational motion into linear motion, such as using toothed belts or rails, as opposed to

Figure 4.5: The phrase ”THANK YOU” as written by SWAN.

using counterweights, fishing wire, and a sliding tray/drawer slider. We would also design a better way to mount our pen so that it would not wobble when writing, and a more elegant pen-up/down mechanism where the pen would contact the paper with just enough force to be able to write, but not so much that it starts rubbing against the paper.
To improve portability we can mount the motors to the main body of the robot, rather than fixing them to the table on which the robot was situated. Using toothed belts or rails as opposed to counterweights would also improve portability.

7. Conclusion

A lot was learned during this project. We learned to be creative, improvise when needed, and use whatever limited resources we had to achieve our objective. We had to overcome many challenges, such as turning 2 motors' rotational motion into linear motion in 2 dimensions, as well as achieving an acceptable level of accuracy and good levels of precision and repeatability using inaccurate motors and a somewhat unstable pen mount. However, in the end, it was very rewarding to see SWAN be able to write different words and phrases.

8. References

- R. Y. Putra *et al*., ”Neural network implementation for invers kinematic model of arm drawing robot,” *2016 International Symposium on Electronics and Smart Devices (ISESD)*, Bandung, Indonesia, 2016, pp. 153-157, doi: 10.1109/ISESD.2016.7886710.
- Á. Hámori, J. Lengyel and B. Reskó, ”3DOF drawing robot using LEGO-NXT,” *2011 15th IEEE International Conference on Intelligent Engineering Systems*, Poprad, Slovakia, 2011, pp. 293-295, doi: 10.1109/INES.2011.5954761.
- R. Smith, ”An Overview of the Tesseract OCR Engine,” Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991.
- Liang, Dt., Liang, D., Xing, Sm. *et al.* A robot calligraphy writing method based on style transferring algorithm and similarity evaluation. *Intel Serv Robotics* **13**, 137–146 (2020). https://doi-org.login.ezproxy.library.ualberta.ca/10.1007/s11370-019-00298-3