Final Report: Pattern-O-Matic 2000

Maya Abo Dominguez
Harvey Mudd College
mabodominguez@hmc.edu

1. Introduction

Many professional and hobbyist sewers use commercial patterns when sewing garments. However, dealing with tissue paper commercial patterns is a cumbersome process and often involves cutting out unwieldy and fragile shapes that are prone to tearing. An example of a commercially available tissue pattern can be seen in Fig. 1. Using printable PDF patterns requires piecing together sheets of paper before even cutting out the pattern pieces.

A digital storage method for patterns that allows sewers to easily cut out the pattern pieces from fabric by projecting the pattern onto the fabric would solve both of these problems.

Segmenting the individual pattern pieces from a large pattern sheet is similar to many existing diagram parsing problems [5], and sewing patterns are rich with information to gather from the pattern, including symbols, text, and contours. This involves using object character recognition (OCR) to detect and decode the text in the PDF, and using classical computer vision techniques to segment out the individual pattern shapes.

2. Method

2.1. Data

I used PDF sewing patterns from Peppermint Magazine[4], which has free sewing patterns available for download. I chose to use PDF patterns to simplify the process, since taking pictures of paper patterns would be unwieldy and expensive. Additionally, the clarity of the PDF images would allow for better segmentation of the pattern shapes and the detection and parsing of the text. I also worked with printshop sized PDF patterns (A0 instead of A4 sized sheets), to avoid having to piece together many smaller images. I focused on a single pattern, the Milton Pinafore seen in 3, since it had some more complex shapes, text at various angles, and dotted outlines of the pattern pieces, while still being a relatively simple pattern.

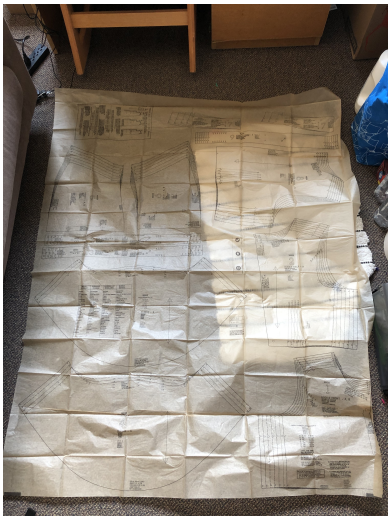

Figure 1. A commercial pattern for a relatively simple blouse laid out on the my dorm room floor. Note the unwieldiness of the tissue pattern.

2.2. Method

My method to take a sewing pattern and information on fabric yardage and process it so it can be projected onto the length of fabric for easy cutting can be broken into three parts.

- Segmentation of pattern pieces from an image of a sewing pattern.
- Parsing information on grainline to orient the pattern shapes
- Using orientation information from pattern, allow users pack pattern pieces into a minimal packing into the fabric yardage.

2.3. Pattern shape segmentation

A PDF sewing pattern sheet often includes many pattern shapes in it. This can be seen in 3. Normally, a sewist would cut out these shapes from the sheet, and based on the information in the pattern, arrange the cut out shapes onto

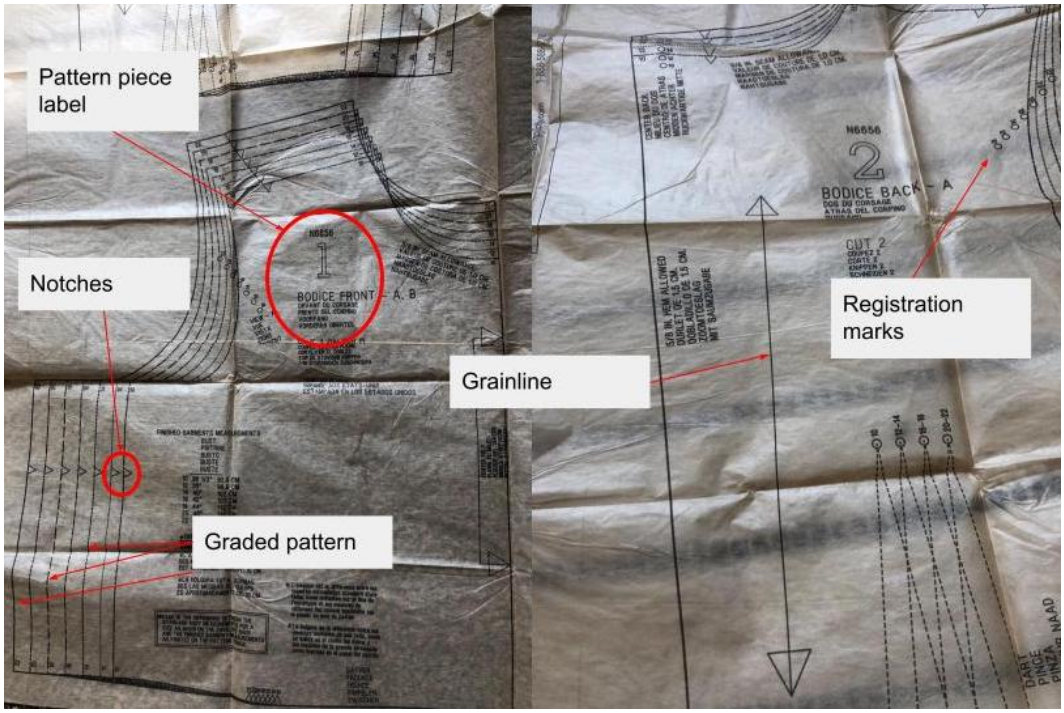

Figure 2. Examples of information present in commercial sewing patterns. Often patterns also include a legend to explain the various symbols, as well as other landmarks, such as a line indicating waist or bust line.

a length of fabric. From there, the sewist cuts out the fabric in the shape of the pattern shapes.

The goal was to recreate this process by segmenting out the individual pattern shapes from the overall sheet. This was a more complicated process than I originally imagined due to the complex shapes the patterns could take and the size grading. This meant that for each pattern shape there were many sizes nested inside each other, with various line styles outlining them, such as dashed or dotted. To be able to get nice contours of each pattern piece, I did some morphological transformations to close the dotted and dashed lines. I dilated once using a 4×4 kernel and then did a closure using a 3×3 kernel over 5 iterations.

From there, I did a simple binary threshold to get a binary image to perform the contour detection steps. Since sewing patterns are just black and white, for the most part, this was very simple. I used the function built into OpenCV `findContours()` to find the outermost contour of each shape. This meant that I would not necessarily segment out my correct size, since the largest size is the outermost contour. I thought that was acceptable since it would allow the sewist to decide which size to cut out at the time of cutting, and would make the problem significantly easier.

After obtaining the outermost contour, I created a mask with each of the pattern shapes, and masked the original pattern sheet, cropping it into the individual pattern shapes. The mask can be seen in fig.4, and an example of an individual pattern shape can be seen fig. 5.

2.4. Detecting Grainline

Once the pattern shapes were segmented, the next step is to detect the grainline of the pattern piece and orient the image correctly. The grainline indicates the correct orientation of the piece with the warp and weft of the fabric, and is vital to having well-fitting garments. In many patterns, the grainline is indicated with a line and the word "grainline."

To detect the grainline of each pattern piece, I used the OCR Python package `EasyOCR` [1] to detect and decode the text present in the image. It uses a pretrained machine learning model to detect the text and feature detection to parse the image to text. However, due to the restrictions of EasyOCR, it only can detect and decode text that is horizontally oriented upwards (oriented regularly). The text detected before and after applying the algorithm can be seen in figures .

Based on this restriction, I determined Algorithm 1 to orient a single pattern piece correctly according to the grainline. This algorithm was run on every pattern shape in a pattern sheet.

Since many of the pattern pieces need to be rotated a multiple of 90 degrees, this works most of the time. However, there are cases where the text is at an an angle that this algorithm cannot catch. In this case, the user would have to use a third-party service to rotate the pattern image.

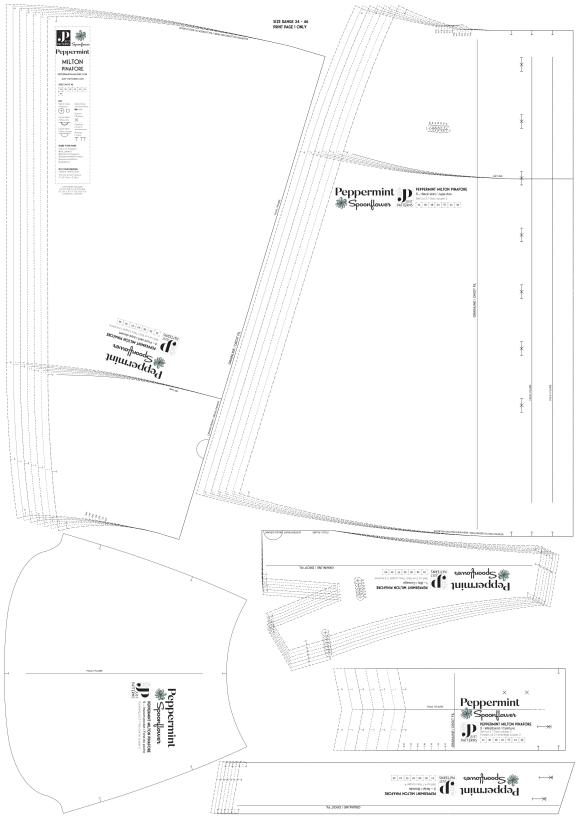

Figure 3. The A0 printing pattern sheet for the Peppermint Magazine Milton Pinafore pattern

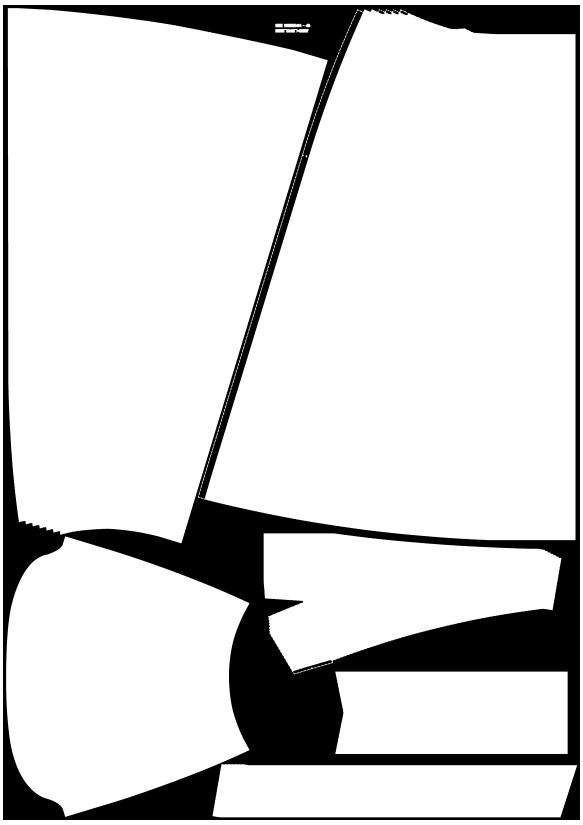

Figure 4. The mask created by using the outermost contours

Algorithm 1 Orient pattern pieces based on grainline

while "grainline" not detected or $rotated < 4$: **do**
  if "grainline" detected in $Image$ **then**
    break
  else
    $Image$ = $Image$ rotated by 90 degrees
  end if
end while

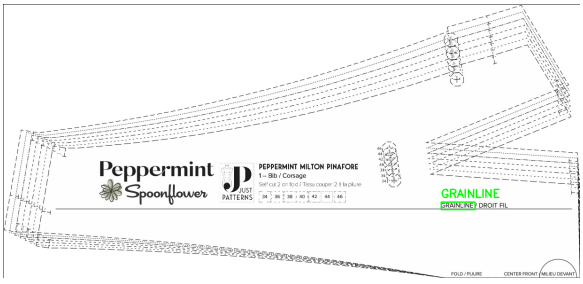

Figure 5. An example of an individual pattern shape, after rotating for the correct orientation based on its grainline. The grainline is highlighted with the green text

2.5. Interactive shape arrangement

The third and final step of the process is to allow the users to arrange the pattern shapes onto a virtual "fabric" such that they fit on the real-world fabric.

Originally, I had planned on using 2D bin-packing algorithms to automatically pack the pattern shapes onto the virtual "fabric." However, this proved to be a very difficult task, since 2D bin-packing is NP-hard [6] here, and even heuristics that attempt to solve it are more computationally intense than is the scope of this project. Since humans are relatively good at this problem, I decided to allow the user to do the arrangement of the pattern pieces. This would also allow the human to review the pattern pieces to ensure that they are oriented correctly and the right pattern pieces are there before committing to an arrangement.

I wanted users to be able to input the dimensions of the fabric (width and yardage) and drag and drop the images of the individual pattern shapes that were scaled to the correct size based on the virtual "fabric" size onscreen. I used the Python GUI package `Tkinter`[2] to create my GUI. To toggle between pattern pieces, the user can left-click, and the selected pattern piece can be dragged and dropped anywhere on the fabric. However, I realized that many reasonable yardages needed for garment creation would fit poorly

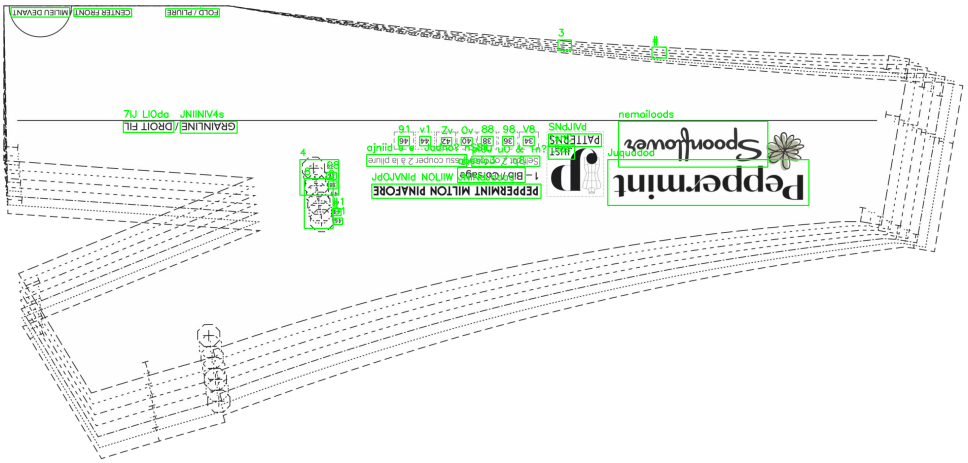

Figure 6. An example of the text detection using `EasyOCR` on incorrectly oriented text. This is what the pattern shapes look like before applying Algorithm 1

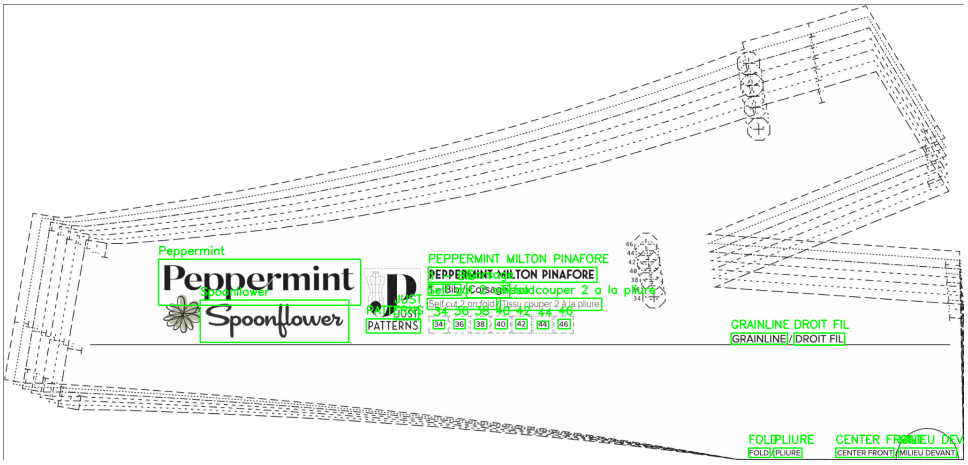

Figure 7. An example of an individual pattern shape with text detection from `EasyOCR`, after applying Algorithm 1 to orient the shape with respect to the grainline.

on a computer screen, I allow for scrolling across the fabric.

To aid in the alignment of virtual "fabric" with the real-life fabric when projecting, I also included a square that should measure 2" $\times$2". A screenshot of the interactive GUI after the user has successfully arranged the pattern pieces can be seen in figure 8

2.6. Projection onto Length of Fabric

Finally, I would like to project the arranged pattern pieces onto the lengths of fabric so they can be easily cut out. One problem is that because the lines of the sewing pattern are relatively thin and detailed, it is really hard to see the lines of the sewing pattern with the resolution of projector I have. I believe that for it to be used regularly, a higher quality projector should be used. One way to work around this could be to just increase the thickness of the lines by dilating, but then we run the risk of losing information in the pattern, such as sewing landmarks or sizing information. This problem is highlighted in figure 9, where I projected the virtual "fabric" onto my wall to get a sense of how it would work when projected onto real-life fabric. However, the concept appears to be feasible.

3. Results

Due to the nature of the project, there are few analytical results. However, I can speak qualitatively to the process. The overall process was successful in being able to segment out the pattern shapes from a pattern sheet and arrange them to be cut out of fabric in real life. The text detection process works remarkably well when the images are oriented correctly, likely because the `EasyOCR` package was designed for document reading, which PDF patterns are similar to.

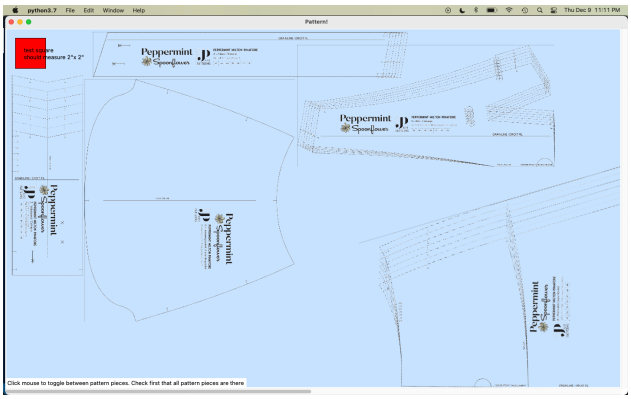

Figure 8. A screenshot of the interactive pattern arrangement GUI. The user can drag and drop individual pattern pieces, toggling between them by clicking. Note the test square in the top left corner.

Figure 9. The interactive GUI projected on my wall. Note that the poor quality of the projector makes it difficult to see the lines of the pattern.

However, it is very slow to detect the text each time. This could pose a problem when refining the rotation algorithm.

The user interface to allow users to drag and drop the pattern pieces is a little clunky to use, and adjustments to the fabric dimensions, scale factor, and image directory need to be changed directly in the code. There is room for improvement there, but it overall works pretty well for its designed purpose.

My project is somewhat comparable to Morgan Donner's experiences using projection for sewing [3]. She just uses the patterns (without any of the rotation) and scales them manually. Her actual projection results are much better, likely due to the quality of her projector.

4. Conclusion

In this project, I attempted to make the pattern using process more usable for sewists who are tired of paper patterns. I think I was mostly successful in being able to segment each pattern piece from a pattern sheet, orient them correctly, and allow the user to arrange them to be cut out. There is definitely room for improvement, but I learned a lot in the process about image processing, text detection and user interface development.

5. Repository

My code can be accessed in the following repository (with sample images): https://github.com/mabodominguez/pattern-o-matic

References

[1] Easyocr. 2

[2] tkinter - python interface to tcl/tk. 3

[3] Projector vs paper patterns - a very quick comparison, December 2020. 5

[4] Peppermint magazine, Aug 2021. 1

[5] Floraine Berthouzoz, Akash Garg, Danny M. Kaufman, Eitan Grinspun, and Maneesh Agrawala. Parsing sewing patterns into 3d garments. *ACM Trans. Graph.*, 32(4), July 2013. 1

[6] Adrian Sy. A fast 2d packing procedure for the interactive design and preparation of laser-cut objects for fabrication, May 2020. 3