ECE 471: Assignment 2 (Part B): Data Analysis

Geena Smith
Faculty of Engineering
University of Victoria
Victoria, Canada
gmsmith@uvic.ca
V00835915

Joshua McIntosh
Faculty of Engineering
University of Victoria
Victoria, Canada
jmcintosh@uvic.ca
V00875715

I. Bias in Data

In the field of computer vision, biases in data are a real concern that must be mitigated when designing any algorithm, especially in symbol spotting algorithms such as the one we are implementing for our project. In order to have a successful project, we need to not only identify the biases in our data set, but also take appropriate steps to limit them.

Our data set consists of an instruction set for a childs dresser, which includes both pictorial and text directions. We chose this data set as our algorithm is focused on detecting both graphical and orthographic symbols in documents, and we felt instruction manuals had a sufficient amount of both. We also felt that the graphical symbols included in the document are drawn in a way that does not have too much detail, yet is not a toy example of a symbol that would be very easy to detect. From the document, we extracted roughly 4 graphical symbols and 4 text words to use for querying, and 15 images for testing and training. An example of some of the querying symbols include a hexagon screw head, a side-view of a screw, a drawer side, and the words Washer and Bolts.

There are numerous biases within our data set that we acknowledge and need to address in order to have a successful implementation. First of which is that our data set suffers from being very consistent. As we are working from a single document, all of the symbols are drawn in a particular style, and each instruction page is formatted in a similar way, with images of the required parts in the upper right quadrant, pictorial instructions in the center, and text description in the lower third of the page. Additionally, the instructions are all digital, so the structure is very regular (eg. straight lines, perfect 90 degree corners) and text is horizontal. This high level of structure and consistency throughout will likely cause our algorithm to be too adapted to our data set, such as is a common problem brought up by Torralba and Efros [1]. We would expect our results to reflect this by having a very high recall rate.

Another source of bias is with the variety of symbols in our data set. The types of graphical symbols seem to fall into a number of closely related groups. For example, there are many images of screws, all varying slightly by the length to width ratio. These images all have a high level of similarity, but very few common features compared to other types of symbols, such as the circular screw heads. The data set used in the original paper consisted of electrical drawings [2]. These electrical drawings had multiple words with common prefixes/suffixes, and many graphical symbols which consisted of various linear sub-structures, but were not highly similar to one another. The original paper had relatively high precision and recall with their data set, and we would expect with our data set to have a very high recall but lower precision due to the more separate types of symbols.

The final bias within our data that we will mention is that of observer bias, or designing the sample set based on what we expect to see [3]. We chose many of our querying symbols with desirable results in mind. For example, we avoided querying many of the large panels that make up the dresser. These panels looked very similar to each other, which would be more difficult for the algorithm to correctly identify, so instead we chose symbols that have very distinct shapes in order to easier evaluate if our algorithm is working. Since we designed the data set with some prejudice, we are expecting our results to be better than if we had chosen symbols at random or had used a different data set.

We recognize that our data contains some degree of bias, and will likely influence the results we get. We felt the benefit of having symbols, both graphical and orthographic, that are well defined and relatively simple would allow us to evaluate if our algorithm gives similar results to the original paper. Our goal, and that of the original authors, is to determine if we can detect both graphical and orthographic symbols within the same document with some level of reliability, not with high accuracy. Our data set gives us a controlled environment, with well-defined shapes and minimal artifacts, to try and recreate the original paper. If we feel that the algorithm is influenced too much by the bias within the data, we can add to our data set by introducing some degree of randomization when choosing querying symbols, or introduce new data sets altogether.

II. Evaluation Metrics

In order to properly measure how effective our implementation is, a proper method of evaluating its performance is required. Similarly to Rusinol and Llados original paper [3], we will be evaluating our algorithm based on precision and recall, where 100 percent precision is having all detected symbols be the desired symbol and 100 percent recall is having all desired words and symbols be detected. Since the expected output of the algorithm is a set of coordinates within the dataset, it will be necessary to compare those coordinates against a set of correct coordinates. For each input symbol we wish to test, a set of coordinates will be made for each matching symbol in the dataset, which we will call the validation set. Ideally, this validation set will be created automatically using other image-matching algorithms, which would make it easy to add new data to the dataset, but for now it will be created manually since our dataset is limited in size. Since the voting structure of our algorithm points to the center of the symbol, all coordinates will be based on the center of symbols. In order to validate our algorithm on precision and recall, a comparison of the algorithms output and the validation set will be made. If the output of the algorithm is within a small window of the validation coordinates, it will be considered to have detected the symbol. The precision of both the algorithms coordinates and the validation sets coordinates as compared to the actual center of the symbol is not important. As long as both are within a short distance of the actual center and each other, the precise value of the coordinates does not matter. This will help negate any error in the validation set and reduce false negatives when validating the algorithm. The actual value of precision and recall will be calculated by the formulas

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

where TP is true positives, FP is false positives, and FN is false negatives [4]. While the ideal performance of our algorithm would be to maximize both precision and recall, recall is the more important statistic to maximize. Since an important factor of the algorithm proposed by Rusinol and Llados was to have a coarse retrieval of words or symbols in real-time [2], it is more important to have all matching symbols appear alongside false positives, allowing the user to manually sort through them, than it would be to miss a true match. This means that if we prioritize recall, the precision of our algorithm will likely decrease [4]. Thus, our algorithm will be judged based on primarily its recall performance with precision performance as a secondary. Beyond error introduced by manually creating the validation set, a possible source of error is having the algorithm detect two symbols close enough together that both are within the window of a validation coordinate. The two metrics above do not account for detecting the same symbol twice, and it will be important that these are caught as to not influence our performance measure. For the current dataset, this is fairly unlikely, as the dataset diagrams are uncluttered and symbols are separated, but if a more difficult dataset is added, this may become an issue.

III. Timeline

The Gantt chart given in Figure 1 is our expected timeline at this point, however we will likely have to make adjustments as we get farther into the project. Through planning, we have identified some potential bottlenecks that we may need to adjust for. First of which is that the main development time will likely fall between weeks 9 and 12 in the semester due to both the midterm and other large assignments due in other classes. Since this is our least busy time of the semester, we intend to get most, if not all of our development done, and allow the last 2 weeks of the semester to wrap up the project and give our oral presentation. If we feel as if our development is taking longer than intended, such as with building the hash structure and proximity graphs which we have determined to be the most complex part of our algorithm, we can use most of the week we have set aside for testing and cleaning our code. We can then move our testing and cleaning to be done in parallel with preparing for our oral presentation,

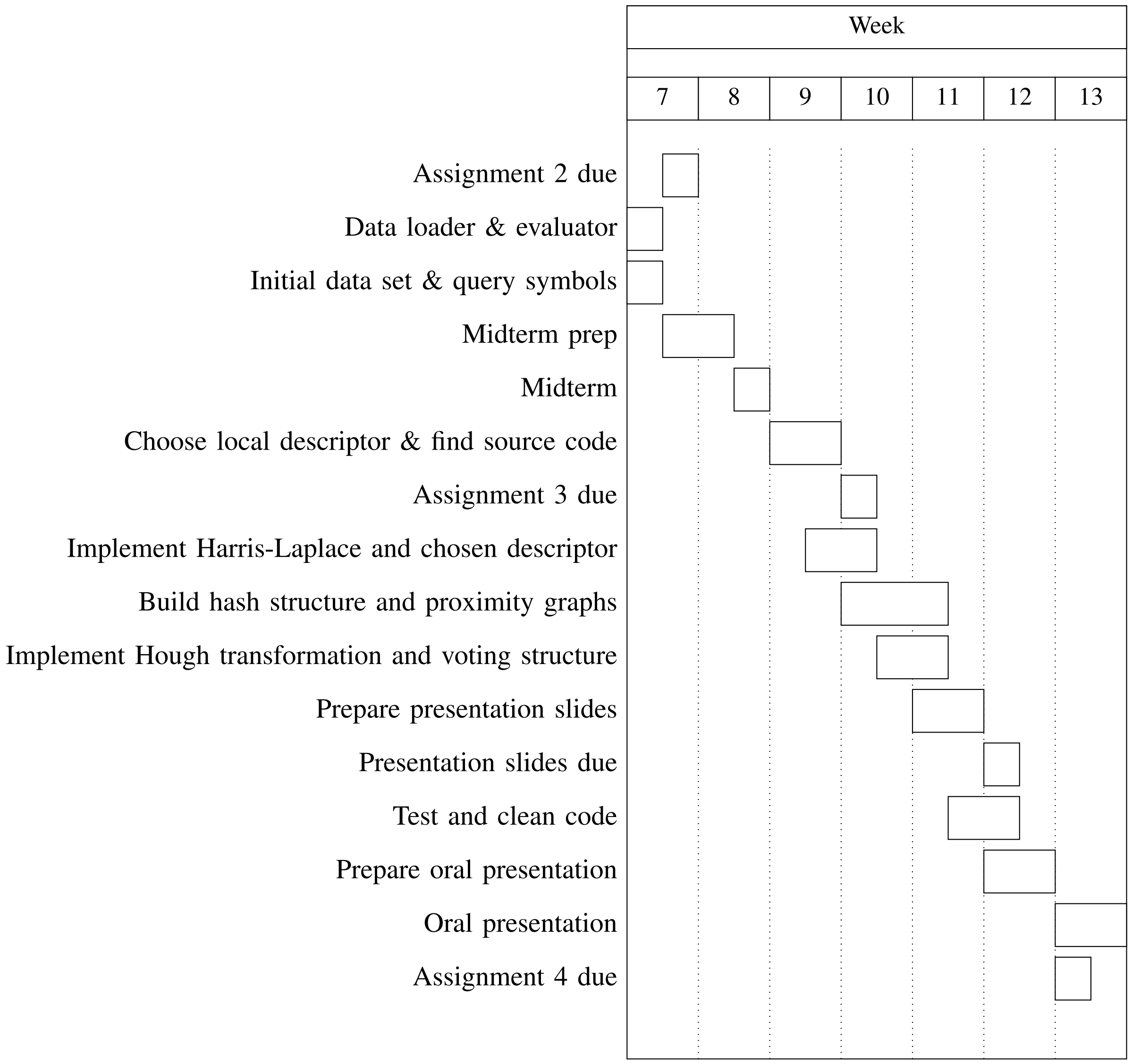

Fig. 1. Gantt Chart

as long as we have enough of our project complete to prepare our slides. Regardless of how easy our project is to implement, we acknowledge that our semester will be busy with other commitments outside of class, especially near the end of the semester, and thus we hope that by front-loading our work we will have the last few weeks for further development if needed.

References

[1] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," *CVPR 2011*, Providence, RI, 2011, pp. 1521-1528.

[2] M. Rusinol and J. Lladnos. "Word and Symbol Spotting Using Spatial Organization of Local Descriptors," *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, Nara, 2008, pp. 489-496.

[3] S. Ghoneim, "5 Types of bias & how to eliminate them in your machine learning project," *Medium*, 16-Apr-2019. [Online]. Available: https://towardsdatascience.com/5-types-of-bias-how-to-eliminate-them-in-your-machine-learning-project-75959af9d3a0. [Accessed: 21-Feb-2020].

[4] "Classification: Precision and Recall — Machine Learning Crash Course," Google. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. [Accessed: 21-Feb-2020].