ECE 471: Word and Symbol Spotting

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. Abstract

In this paper we present our implementation of the method first described by Marcal Rusinol and Josep Llados in their paper entitled *Word and Symbol Spotting Using Spatial Organization of Local Descriptors* [1]. Their novel approach aims to allow for fast and efficient querying of both word and graphic symbols within a set of documents. Although the original paper used a data set consisting of wiring diagrams, we chose to use a data set consisting of assembly instructions for a children's dresser. We felt the symbols included in the data set were a good mix of both word and graphical symbols, and they would be slightly more complex than a toy data set, allowing us to focus on the implementation of our program rather than the data set.

Rusinol and Llados' approach, and that which we chose to follow, relies on first calculating the keypoints and descriptors for the set of images that will build the database to be queried. These descriptors are used to insert keypoints into a hash table in order to facilitate fast querying. Using the descriptors for the query images, keypoints of interest are extracted from the hash table and their spatial arrangement is computed using a proximity graph. A voting system is used to determine the most probable locations of the symbol being queried, and these are evaluated using the labels associated with each data image. Although we did not have a completely working solution at the time this paper was written, we describe our implementation in detail as well as the areas we identified as needing improvement.

II. Introduction

The paper we decided to base our project on, Word and Symbol Spotting using Spatial Organization of Local Descriptors, by Marcal Rusinol and Josep Llados, proposes a solution to identify both word and symbol spotting within a single document. As many documents are converted from paper to digital form, there is a need to improve the accessibility of the information in ways such as being able to browse or modify the documents.

In documents containing mostly textual information, there are existing optical character recognition (OCR) softwares that can be applied to convert raw images into text, and the resulting files can easily be navigated and edited. However, OCR's often have specific constraints on the layout of the raw text and do not work well on graphical symbols.

Symbol spotting was introduced by Tombre and Lamiroy and consists of indexing a large collection of graphical documents based on the symbols that appear in it [2]. Since the time this technique was introduced, symbol spotting has been an emerging topic of research, however there is a similar problem as with text conversion in that symbol spotting does not usually handle both text and graphical symbols without previous separation, and many implementations are computationally expensive.

As will be explained in the following literary review section, word and symbol spotting have been areas of research for a long time, but rely on fundamentally different approaches making spotting of both within the same document very difficult without first separating. The few approaches that do not require separation are either computationally expensive, or make assumptions that may not be applicable to all documents, such as that symbols would fall in certain areas of interest. The approach of Rusinol and Llados, and that which we will be implementing, is addressing these problems. The proposed approach uses a hash table for fast retrieval, and relies on the spatial organization of local descriptors to handle both words and symbols at the same time.

III. Literary Review

The field of both word and symbol spotting has a large number of published works. Many of these works that are applicable to the problem domain we are trying to solve have drawbacks that we feel our implementation addresses.

For example, in the word spotting domain Kuo and Agazzi proposed a method using a hidden markov model [3]. This method works well for poorly printed documents, but requires a learning step to train the model and every word that wants to be queried has to be learned. Llados and Sanchez also proposed a spotting technique based on shape context descriptors [4]. A voting strategy is used to retrieve zones of a document containing a keyword. These techniques are proven to work well, but like most literature require an initial word segmentation and rely on a cross-correlation approach which does not allow for a whole document to be used.

In the symbol spotting domain, the main problem is that the existing solutions are computationally expensive. Some

approaches use rough segmentation to separate symbols before querying, but this is not suitable for large data sets [5], [6]. Other approaches rely on a graph-based representation of the documents, but this is not practical for large sets of data and is also computationally expensive [7]–[9]. Some techniques address the computationally expensive aspect of these approaches, but oftentimes rely on assumptions such as that symbols always fall into regions of interest [10]–[12]. They are also affected by noise or occlusion. Since we can't make assumptions about where symbols will appear, these are not suitable either.

The algorithm described by the authors, and replicated by us, was inspired by the work of Nakai, Kise and Iwamura [13], [14]. In their work they introduced a method to retrieve whole document images from a large database using the spatial organization of keypoints. This addresses the main issues with the previously discussed literature. Both words and symbols can be queried at the same time without previous segmentation, and large data sets can be queried much faster. Where our implementation differs, and ultimately makes it unique, is that we are indexing subparts of a document rather than the document as a whole which allows for a greater range of uses.

The rest of our paper will be focused on our implementation and results. First we will describe our proposed approach in detail, outlining the algorithm and how it differs from Rusinol and Llados' paper. Next, we describe our dataset used and evaluate the results we obtained. Finally, we finish with a conclusion to reiterate the problem domain we are addressing, our approach, and some final thoughts on our work.

IV. Proposed Approach

As stated in the original paper, most object recognition models follow three basic steps. First, points of interest are extracted and taken as keypoints (or primitives as they are called in the paper). The local descriptor is also calculated at each keypoint. Next, matching between the descriptors of the object (the query) and the image is performed. Lastly, a voting scheme is used to validate potential locations and find the objects. The original paper, and thus our implementation, follows this approach.

Initially, our algorithm reads in the set of document images, the query symbols, and labels which are used for our evaluation. For both the images and query symbols, the keypoints are determined. In the original paper, the authors used Harris-Laplace to initially retrieve points of interest, however since we determined we would use SIFT in calculating our local descriptors, we chose to use only SIFT to calculate the keypoints instead. The authors tested multiple off-the-shelf descriptors such as SIFT, shape contexts, Hu's geometric moment invariants, and a set of steerable filters. We chose to only use one descriptor for simplicity, and we determined SIFT would be the easiest to implement as it is available in Python's OpenCV library.

After computing the descriptor for each keypoint, we placed the descriptors for the images into a hash table. The authors did not provide much detail about how they implemented the hash table. For an object O, the keypoints were denoted as $O = p_1, ..., p_n$, and at each keypoint the descriptors were denoted as $d(p_i)$. A hash function, $h(x)$, quantizes the descriptor into an index kj such that $h(d(p_i)) = k_j, i \in [1, n], j \in [i, m], m \leq n$. At each index kj of the hash table, a list of keypoints, $L_j = [p_1, ..., p_r]$, with the same index and similar description will be stored. This process is shown in Figure 1.

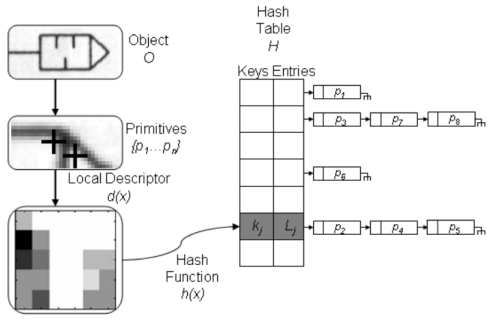

Fig. 1. Rusinol and Llados' general process used to build the hash table [1]

The authors did not give any detail on the size of the hash table relative to the number of keypoints. We chose to make the hash table 75% of the minimum number of keypoints within a single image as we thought this would allow for many keypoints to have the same index, hopefully leading to higher recall rates. The other modification we made was with the data that is stored in each entry of the hash table. In the paper, the authors stored only the keypoints. We decided to store the keypoints, descriptor, and information about which image the keypoint belongs to as this would simplify the implementation of the rest of our algorithm.

The spatial orientation of each symbol was represented using a proximity graph. A graph $G(O) = (V, E)$ was constructed and for each keypoint $v_i \in V$, an edge was created to its k nearest neighbors, with $k = 5$ chosen due to the paper doing the same. Thus, we gain information on the spatial organization of local descriptors by linking them with their closest neighbors.

Once the hash table and proximity graph is built, we could begin querying the data set for specific symbols. A pair of keypoints (k_i, k_j) representing an edge in the proximity graph are retrieved, and similar keypoints are retrieved using the hash table. The combination of two keypoints having similar descriptors and spatial organization to the queried pair of keypoints accumulates evidence which can be used to locate similar examples. A visualization is given in Figure 2.

With a specific query, an algorithm similar to a generalized Hough transform is used as described by Rusinol and Llados, allowing for detection of arbitrary shapes. With each pair of keypoints, a hypothetical center, hC, is calculated that is used to accumulate votes for potential symbols. However, Rusinol and Llados do not specify the details of this algorithm and this had to be implemented manually as an existing algorithm was not found which matched our needs. Specifically, our

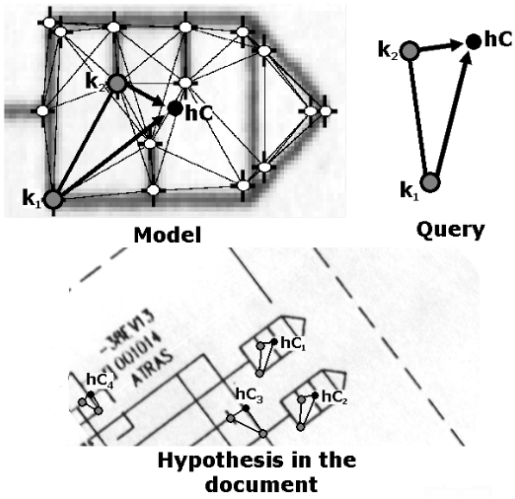

Fig. 2. Rusinol and Llados' example of using pairwise queries to accumulate evidence. [1]

algorithm calculates the angle and magnitude between the points (k_i, k_j) and the center point of the query image, which can then be used to translate the hC to similar pairs of matching descriptors found in the documents. This approach allows the detection algorithm to be invariant to any scale or rotational changes.

The voting space is represented as a dictionary, with a tuple of coordinates (x, y) used as the key and the number of votes as the value. As new points accumulate votes, the coordinate tuple is added to the voting space dictionary, and subsequent votes to that point increment its value.

This voting mechanism will accumulate evidence in areas of similar descriptors and spatial arrangement to the query symbol in question. The areas that likely indicate the query symbol will have a peak in the voting space. After identifying probable locations of our query symbol, we validated our results using our labels that matched our data images and calculated precision and recall values to evaluate our algorithm as a whole.

Unfortunately, our implementation could not be completed as planned and the problems we faced are explained in the next section.

V. Evaluation & Datasets

In the original paper, Rusinol and Llados used a data set consisting of automotive wiring diagrams. In order to make our project unique, a different data set was chosen. Our data set consisted of images taken from an instruction manual for a children's dresser [15]. An example image can be seen in Figure 3. The main reason for choosing the instruction manual for our data set was that it contained numerous words and graphical symbols on the same page, and many of the symbols were present on multiple pages. The symbols were also drawn in the same way and all text was in the same typed font. This meant we would only have to account for variations in scale and could focus on the implementation of our algorithm rather than our data set.

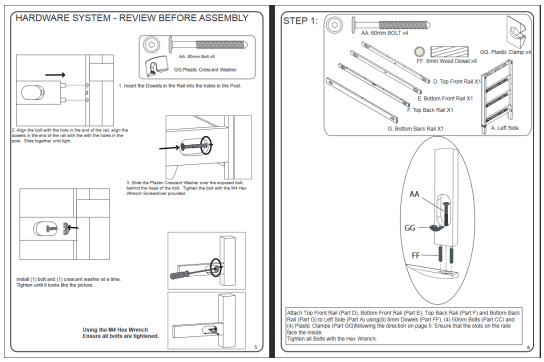

Fig. 3. Sample instruction manual used in the dataset. [15]

In total, our data set consisted of 12 data images of size 1654 by 2197 pixels, and 9 query images taken directly from our data set. Some examples of our query symbols can be seen in Figure 4. We also created label files for each data image which contained the x,y coordinates for the center of each query symbol present. These label files would be used to validate the results of our implementation.

Fig. 4. Examples of query symbols used in our implementation.

Unfortunately, our implementation faced issues during development and was unable to replicate the results given by Rusinol and Llados. Two major problems were identified: first, the number of votes being collected was far too low with only 1 or 2 votes being collected for each data image despite the queried object being present multiple times within the data image; second, the points where votes are being collected are do not correspond with correct locations of queried images within the data image. Figure 5 shows the discrepancy between the accumulated votes and the data image.

The source of these problems were narrowed down to three possible places: either our implementation of the hash table did not work as the paper described and was not returning similar descriptors, the angle and magnitude calculations for getting the hypothetical center of a pair of keypoints was incorrect, and/or applying said angle and magnitude to similar keypoints to accumulate votes was incorrect. First, if the hash table was not correctly returning similar descriptors, then the set of returned descriptors for each of the pair of keypoints were unlikely to be connected in the proximity graph of the data image, and were more unlikely still to be correct. This would explain both problems of lack of votes and incorrect votes, as keypoints not connected in the proximity graph are automatically discarded. Second, if either the calculating or

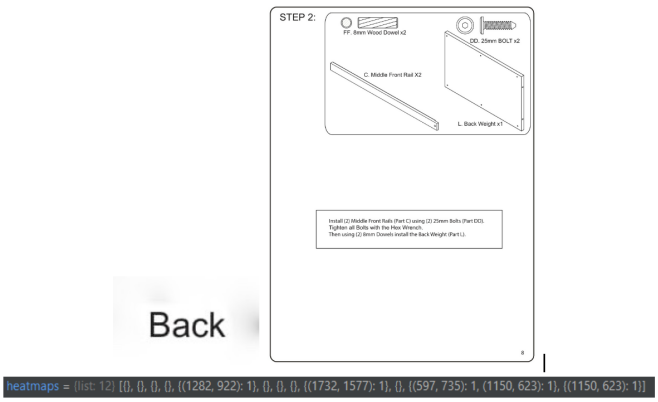

Fig. 5. The query image containing the word "Back", the data image corresponding to the fifth heatmap dictionary, and voting space dictionaries (here as heatmaps). Notice that only 1 vote is accumulated, and the location (1282, 922) does not correspond to the location of the word "Back".

application steps for the angle and magnitude were incorrect, we would expect that each vote would be cast in incorrect places, creating a situation where either votes are scattered throughout the image space forming no collective agreement on a location, or a peak in votes is accumulated in an incorrect location. However, due to a lack of returned votes, it was difficult to discern whether this was an issue. Finally, it is possible that the problem lies elsewhere, such as an aspect of Rusinol and Llados' paper being misinterpreted or not accounted for in our algorithm, resulting in key elements of our algorithm working incorrectly.

VI. Conclusion

This report described our implementation of the method proposed by Marcal Rusinol and Josep Llados in their paper entitled Word and Symbol Spotting Using Spatial Organization of Local Descriptors. Their method aimed to efficiently query documents containing both word and graphical symbols without first having to separate the two types of symbols. By using a hash table to retrieve similar keypoints, the method addresses the computationally expensive aspect of other existing approaches which was described in the literary review section. Although the approach does not aim to have perfect accuracy, its ability to scale to large data sets make it promising for document searching.

Our implementation followed Rusinol and Llados' very closely. We began with reading in our data set and calculating the keypoints and descriptors using SIFT from python's OpenCV library. The keypoints and descriptors of the data set were placed into a hash table using the hash value of the descriptors. Using the query symbols, similar keypoints and descriptors were retrieved from the hash table, and a proximity graph was created to connect each keypoint with its nearest k neighbours. By creating the proximity graph, information about the spatial arrangement could be combined with the descriptors. Given a query symbol, a generalized Hough transform was used to calculate the hypothetical centers between each pair of keypoints and establish a voting mechanism to find potential areas that the query symbol may appear. Areas with a peak in the voting space determined locations of the symbol within the data images, and these areas were combined with the label set to validate our results using precision and recall values.

Although we had a good understanding of our algorithm, our implementation did not manage to replicate the results of the paper within the time span of this project. We identified three potential areas that we had gone wrong. First is that our hash table was not working as planned, meaning we would not extract the keypoints that we expected. The second possible area was in the calculation of our hypothetical centers, hc, that may have resulted in the wrong values being calculated. Lastly, we may not have applied the hypothetical center values to our data images correctly. Given more time and experience with this field or research, it is likely that our algorithm would be able to replicate the promising results of Rusinol and Llados' paper.

In implementing Rusinol and Llados' method, we were able to get hands-on experience with much of the content we learned in lecture. It also allowed us to identify areas we would change or improve if we were to start again. Aside from fixing the areas we identified to have a fully working solution, future work on this project could include testing with different and more challenging data sets, or testing with very large data sets. Overall we felt our results were promising despite not having a working solution, and the project allowed us to demonstrate the knowledge we gained this semester.

References

[1] 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*, 2008.

[2] K. Tombre and B. Lamiroy. Graphics recognition - from re-engineering to retrieval. In *Seventh International Conference on Document Analysis and Recognition, ICDAR*, pages 148155, 2003.

[3] S. Kuo and O. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-d hidden markov models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(8):842848, 1994.

[4] J.Llados and G.Sanchez. Indexing historical documents by word shape signatures. In *Ninth International Conference on Document Analysis and Recognition, ICDAR*, pages 362 366, 2007.

[5] S. Tabbone and L. Wendling. Recognition of symbols in grey level line-drawings from an adaptation of the radon transform. In *International Conference on Pattern Recognition, ICPR*, pages 570573, 2004.

[6] S. Tabbone, L. Wendling, and K. Tombre. Matching of graphical symbols in line-drawing images using angular signature information. *International Journal on Document Analysis and Recognition*, 6(2):115125, 2003.

[7] E. Barbu, P. Herroux, S. Adam, and S. Trupin. Frequent graph discovery: Application to line drawing document images. *Electronic Letters on Computer Vision and Image Analysis*, 5(2):4757, 2005.

[8] J. Llados, E. Mart, and J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(10):11371143, 2001.

[9] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In *Graphics Recognition Methods and Applications*, LNCS 1072, pages 123134. 1996.

[10] P. Dosch and J. Llados. Vectorial signatures for symbol discrimination. In *Graphics Recognition*, LNCS 3088, pages 154165. 2004.

[11] M. Rusinol and J. Llados. Symbol spotting in technical drawings using vectorial signatures. In *Graphics Recognition. Ten Years Review and Future Perspectives*, LNCS 3926, pages 3546. 2006.

[12] D. Zuwala and S. Tabbone. A method for symbol spotting in graphical documents. In *Document Analysis Systems VII*, LNCS 3872, pages 518528. 2006.

[13] T. Nakai, K. Kise, and M. Iwamura. Camera-based document image retrieval as voting for partial signatures of pro- jective invariants. In *Eight International Conference on Document Analysis and Recognition, ICDAR*, pages 379 383, 2005.

[14] T. Nakai, K. Kise, and M. Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In *Document Analysis Systems VII*, LNCS 3872, pages 541552. 2006.

[15] *6 Drawer Dresser Assembly Instructions*, Delta Children, New York, NY, 2014. Accessed on: January, 30, 2020. [Online]. Available: https://images-na.ssl-images-amazon.com/images/I/C13t%2B1TH9tS.pdf