Automated Text-Image Comic Dataset Construction

Maciej Styczeń
m.styczen@student.tudelft.nl
under supervision of:
Lydia Chen & Zilong Zhao

I. Introduction & problem statement

Comic illustration-transcription pairs form an interesting dataset for tasks such as:

- comic illustration synthesis
- face recognition on comic characters
- humor detection in comic dialogues
- automated comic translation

Challenge: Dataset creation becomes a bottleneck: to get illustrations with text, each comic strip has to be divided into panels and transcribed. Not feasible to do it manually on a large scale.

Fig 1. Text-image pair creation. Strips need to be downloaded and divided into panels, the dialogues need to be transcribed.

II. Method

Solution: Propose an automated text-image comic Dataset Construction Pipeline (DCP), consisting of 3 stages:

1. Web scraper: automatically download all the images
2. Panel extractor: segment the image into panels
3. Text extractor: extract the text using OCR with additional processing:
    - Pre-processing: up-scaling, binarization
    - Post-processing: clustering-based text ordering and output autocorrect

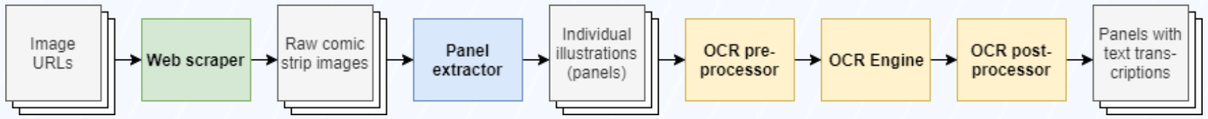

Fig 2. Automated dataset creation pipeline.

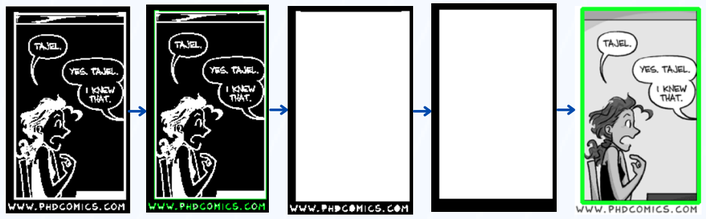

Fig 3. The steps of the panel extraction procedure. From left: binary comic image - outermost contours - filled contours - noise removed - proposed panel bounding box.

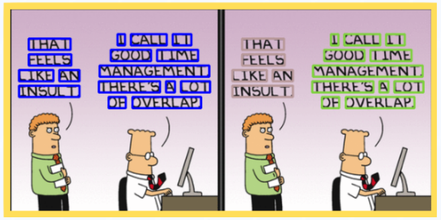

Fig 4. Example of bounding box clustering. The original bounding boxes (left) and the identified clusters (right)

III. Experiments & results

Experiments on three comic series: ***Dilbert, PHD Comics, Garfield***

1. Web scraper:
    - Successfully scraped 28000 comics
    - 149ms per comic on average
2. Panel extractor:
    - Calculate **Intersection over Union [1]** between detected and ground-truth, success if at least 90% overlap
    - Compared with baseline: **Kumiko [2]** comic cutter, see **Table 1**
3. Text extractor:
    - Calculate **Normalized Levenshtein Distance [3]** between true and detected transcriptions
    - Evaluate the impact of adding pre- and post-processing steps, finding the best combination
    - Test with Tesseract [4] and Cloud Vision API [5]
    - results presented in **Table 2** and **Figure 5** obtained using Vision API

Method	Success rate panel	Success rate strip	Time per image
Kumiko (baseline)	93%	82%	656ms
DCP (our method)	97%	89%	1.5ms

Table 1. Results of the panel extraction evaluation.: success rates on panel and strip levels and the average processing time per comic. Tested on a dataset of 1100 manually marked panels.

Experiment	Segmentation	Pre-processing upscale	Pre-processing binarize	Post-processing cluster	Post-processing autocorrect	Avg. error
#1	✗	✗	✗	✗	✗	0.526
#2	✓	✗	✗	✗	✗	0.105
#3	✓	✓	✗	✗	✗	0.102
#4	✓	✓	✓	✗	✗	0.103
#5	✓	✓	✗	✓	✗	**0.075**
#6	✓	✓	✗	✓	✓	0.098

Table 2. Results of evaluating pre- and post- processing techniques for text extraction. Error represented by normalized Levenshtein distance.

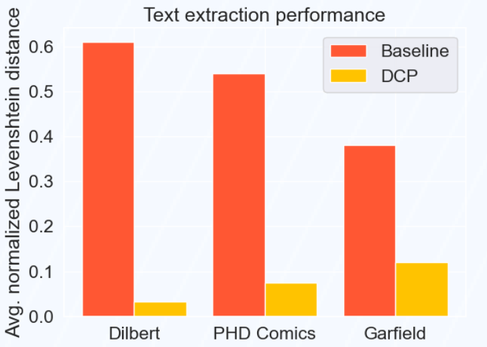

Fig 5. Comparison of text extraction performance between baseline (out-of-the-box OCR) and best results of DCP (with pre- and post- processing).

IV. Discussion & conclusions

- Web scraping performed flawlessly, downloading all comics at a high pace.
- Panel extraction algorithm detected 97% of the panels correctly, outperforming the baseline approach in terms of both accuracy and speed.
- Adding segmentation, pre- and post-processing steps to the OCR pipeline decreased the error 7 times compared to the out-of-the-box OCR.
- Up-scaling the input image had a positive impact on performance, but binarization did not
- Clustering-based output ordering reduced the error by 25%, but the autocorrect step had a negative impact, mostly due to comics containing onomatopeias and exclamations, that are not present in dictionaries.
- Overall the pipeline processed most comics correctly, but a noticeable amount of errors still appeared, mainly at the text extraction stage.

References:

[1] Jaccard index. May 2021.URL: https : / / en . wikipedia .org/wiki/Jaccardindex.
[2] Kumiko. URL: https://github.com/njean42/kumiko/
[3] Li Yujian and Liu Bo. "A normalized Levenshtein distance metric". In: IEEE transactions on pattern analysis and machine intelligence 29.6 (2007), pp. 1091–1095.
[4] Tesseract. URL: https : / / github . com / tesseract - ocr / tesseract.
[5] Google Vision API. URL: https:// cloud . google . com/ vision/docs/ocr.

Comic strips from

- Dilbert by Scott Adams, dilbert.com
- PHD Comics by Jorge Cham, phdcomics.com.

Bachelor thesis for:
BSc Computer Science & Engineering @ TU Delft