Automated Text-Image Comic Dataset Construction

Maciej Styczeń
under supervision of
Lydia Chen, Zilong Zhao
Delft University Of Technology

***Abstract*—Comic illustrations and transcriptions form an attractive dataset for several problems, including computer vision tasks, such as recognizing character's faces, generating new comics, or natural language processing tasks like automated comic translation or detecting emotion in the dialogues. However, despite a large number of comic strips published online, very few datasets of annotated comic illustrations are available. This forms a bottleneck for further advancements in the field. The source of the data scarcity is the manual labor required for annotation — one has to download the comic strips, separate each strip into panels (individual illustrations), and transcribe the text. Automating the process is needed, but it poses several challenges. Panel detection in comic strips is non-trivial, due to varying layouts and styles of comics. Automated transcription is also challenging, as the out-of-the-box optical character recognition (OCR) models struggle with diverse fonts, hand-writing styles, and backgrounds. We design an automatic comic text-image dataset construction pipeline, termed DCP, consisting of three components: (i) web scraping, (ii) panel extraction, and (iii) text extraction. A multi-threaded comic scraper is created to download all the comics. A panel extraction algorithm based on panel frame detection is developed to divide the comic strips into individual illustrations. Lastly, to effectively extract the text using OCR, we propose additional pre-processing and post-processing steps, namely, up-scaling and binarizing images, clustering-based text ordering, and dictionary-based autocorrect. We extensively evaluate the prototype of DCP on three comic series: *PHD Comics*, *Dilbert*, and *Garfield*. Web scraping is used to downloading over 25000 comic strips at an average pace of 149ms per image. Panel extraction results on 1118 panels show success rates of 100%, 97%, and 71% for *Dilbert*, *PHD Comics* and *Garfield* respectively, outperforming the baseline in terms of accuracy and speed. The text extraction algorithm, tested on 1100 comics, achieves a 7x error reduction compared to the out-of-the-box OCR.**

I. Introduction

Comic series like *Garfield* [1], *PHD Comics* [2], or *Dilbert* [3], contain thousands of comic strips, forming an attractive dataset for several experiments. For example, current face detection and recognition systems consistently perform well on human faces [4]. The same models, if appropriately trained, could prove equally successful when applied to comic characters' faces. The introduction of Generative Adversarial Networks (GANs) [5] sparked rapid advancements in the field of image synthesis, including generating images based on a descriptive piece of text [6, 7]. Such GAN models can be used to research synthesizing comic illustrations based on their transcriptions. Natural language processing tasks, such as

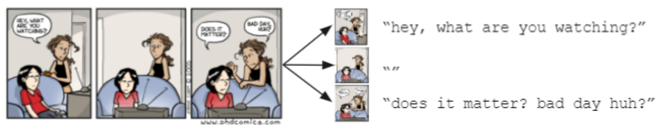


Fig. 1: Input (left) and output (right) of the dataset creation process. Comic from "Piled Higher and Deeper" by Jorge Cham

humor detection [8], and automated translation could also be applied to comic dialogues. However, to be able to perform those innovative experiments, a large, high-quality dataset is required.

Data collection and preparation is a significant bottleneck in machine learning [9]. Obtaining enough high-quality training samples is necessary for the success of machine learning systems [10]. However, manual preparation can be tedious for large datasets that require complex pre-processing. That is precisely the case for the comics data — to construct a text-image comic dataset, a multi-phase preparation procedure is required. The comics strips need to be downloaded from the web, then each comic strip should be separated into individual illustrations (panels), and the text transcription has to be extracted (see Fig. 1). Performing that preparation procedure manually for tens of thousands of comic strips is unfeasible, therefore automating it would be valuable.

The purpose of this paper is to design, implement, and evaluate an automatic comic text-image dataset construction pipeline, termed DCP, consisting of three components: (i) web scraping, (ii) panel extraction, and (iii) text transcription, see Fig. 2. To achieve that, current state-of-the-art web-scraping, image segmentation, and optical character recognition (OCR) techniques are evaluated, adapted, and combined. More specifically, the goal is to answer the following research questions:

1) How to efficiently scrape the image data from comic websites?
2) How to create a panel extraction algorithm that can deal with varying comic layouts?
3) How accurately can the state-of-the-art OCR tools extract the text from comic strips?
4) What pre-processing and post-processing techniques can be applied to the images to improve the performance of the OCR models?

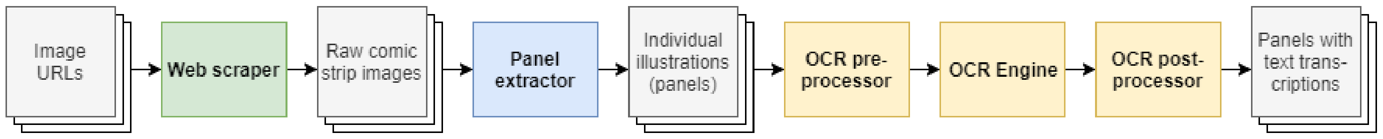


Fig. 2: An overview of the automated dataset creation pipeline.

To automatically perform the dataset creation without sacrificing data quality, one needs to overcome several challenges. Comic panel extraction is a difficult task, as the layouts and drawing styles of comics vary significantly. Evaluation of the existing segmentation methods, for example using the intersection-over-union score, is needed to find one that performs consistently on comic illustrations. Transcribing comics using OCR is also challenging, as the current OCR models struggle with varying hand-writing styles and backgrounds [11]. This research attempts to improve the OCR accuracy by pre-processing the input using image binarization with adaptive thresholding [12] and up-scaling. Another difficulty is the correct ordering of the words output by the OCR. By default, the output is ordered in a line-by-line manner, but for comics also the panel-by-panel, and bubble-by-bubble orderings have to be considered. Text extraction is performed at the panel level and optionally concatenated to obtain full comic transcription. Hierarchical clustering [13] is applied to group all bounding boxes belonging to the same speech bubble and determine the right ordering.

Section II covers the related work and presents the current state-of-the-art in the domain. Section III describes the methodology behind this paper's contributions: it introduces the design of the pipeline, describes in detail the created panel extraction algorithm, and presents the solutions developed to improve the performance of the OCR. Section IV gives insight into conducted experiments and their results, aiming to give a detailed evaluation of each of the steps of the DCP. Section V discusses the results and gives ideas for future work on the topic, and Section VI reflects on the ethical implications and reproducibility of the findings.

II. Related work

We structured the related work in the following categories: work regarding automated dataset creation, web scraping techniques, proposed panel extraction methods, and text recognition in comic strips.

Existing literature shows success in automated dataset building, including web-scraped image datasets, such as *FaceScrub* [14]. However, automated pre-processing of comic strip images remains an unexplored territory. Furthermore, no publications on automated dataset creation involve image segmentation and text extraction steps in the pipeline. This paper aims to explore this gap.

Web scraping has become common and achievable — with the support of modern programming languages and libraries, it is possible to retrieve and process the contents of an arbitrary static website, such as the comic strip websites [15].

Panel extraction has been studied widely, mostly in the context of applying it to mobile comic viewing apps [16]. Some solutions use convolutional neural networks (CNN) to train an object detection model that can find panel positions in a comic [17]. However, this approach would most likely require additional training to perform well on unseen comic series, therefore it is not easily applicable to this papers' problem. Other methods utilize image processing techniques, such as mathematical morphology and region growing for finding the background and extracting the panels [18, 19, 20]. Those solutions are better suited for our use case and they form a basis for the developed panel extraction algorithm.

Current text recognition solutions have a very high accuracy at identifying and extracting text but still have some limitations, including dealing with colored backgrounds, small fonts, and handwritten text [11]. Unfortunately, those are all present in comics — comic strips are characterized by high variability of fonts and styles, often along with complex backgrounds, noise, and low resolution. To overcome those difficulties, researchers propose domain-specific training of OCR models [21] and output post-processing, such as text validation and correction [22]. The reading order of a comic is not straightforward either: comics are read panel after panel, bubble after bubble, unlike most documents, where top-to-bottom, left-to-right ordering is sufficient. Overall, comics form a challenge for OCR, and current engines cannot deal with it out of the box.

III. DCP: Dataset Construction Pipeline

This section describes in detail the techniques used to build the automated dataset construction pipeline. First, the system design and its components are presented. Then, for each step of the DCP pipeline, the proposed method is explained and motivated.

A. System design

The system is designed as a pipeline consisting of three components: (i) web scraping, (ii) panel extraction, and (iii) text extraction, see overview in Figure 2.

The input for the system is a list of URLs to raw comic images. The URLs are passed to the first stage of the pipeline: the web scraper. The scraper downloads all the comics and stores them locally. Then, the full comic strips need to be segmented into individual illustrations: the images are fed to the panel

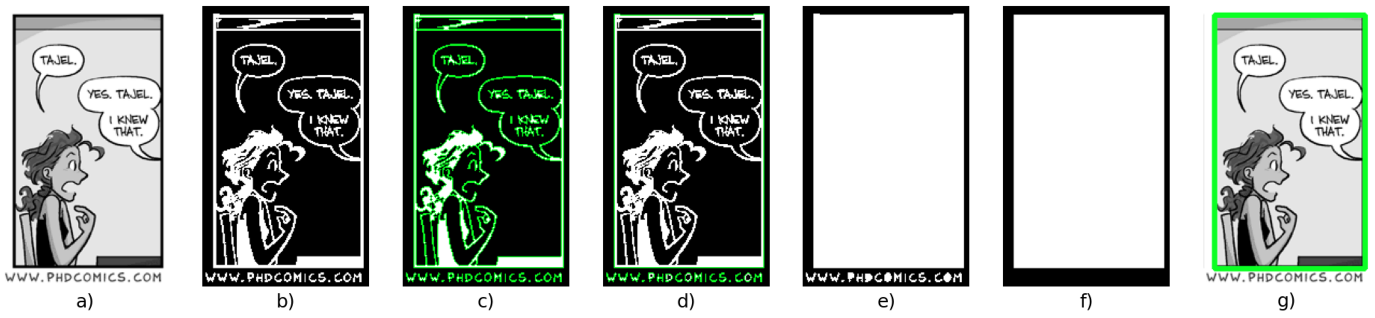


Fig. 3: A visualization of the steps of the panel extraction procedure, for simplicity, presented on a single panel example. The same steps apply to a strip with multiple panels. From the left: a): the original image in grayscale, b): binarized image c): all the contours identified in the image, d): the outermost contours, e): the outermost contours filled with white color, f): noise removal using morphological opening g): proposed panel bounding box. (Comic strips from "Piled Higher and Deeper" by Jorge Cham www.phdcomics.com [2])

extraction stage, where each full comic is divided into panels, and the panels are saved to disk. Finally, to perform automated text transcription, the illustrations are fed into the OCR stage. First, pre-processing is applied to the images, to make the text easier to detect, then the illustrations are sent to the OCR engine, and lastly, the OCR output is post-processed to reduce the error. The output of the system consists of the individual panels, along with their transcriptions, or optionally, the original, full comics with their transcriptions obtained via concatenating the individual panels' transcriptions.

B. Downloading the comics

Downloading the comic images can be achieved using the following scraping procedure:

1) For each comic website, list the URLs of all the comic images.
2) For each URL, send an HTTP GET request.
3) For each response, extract the image and save it to disk.

Performing these operations on a single thread does not scale well, as, for each URL, the process would have to wait for the server's response, before it can send the next request. Therefore, multi-threading is utilized to improve efficiency — several threads send requests and deal with the responses in parallel achieving significant speedup.

There exists extensive library support for web scraping for most popular programming languages. For instance, the process can be implemented easily using Python's *requests* and *BeautifulSoup* libraries. The solution for this task can be reused for an arbitrary comic website — the only additional work when introducing a new comic source is getting the list of all the image URLs.

C. Panel extraction

The panel extraction process leverages the presence of frames around comic panels to perform the segmentation. An overview of the process is presented in Fig. 3.

In the first two steps, the image is converted to grayscale and binarized using adaptive Gaussian thresholding, as visualized Fig. 3a-b. Adaptive thresholding establishes the threshold value separately for each pixel based on its neighborhood, resulting in less noise than global thresholding and preserving the contours and features in the image. Usually, this results in an image that is easier to analyze [23], see Fig. 4.

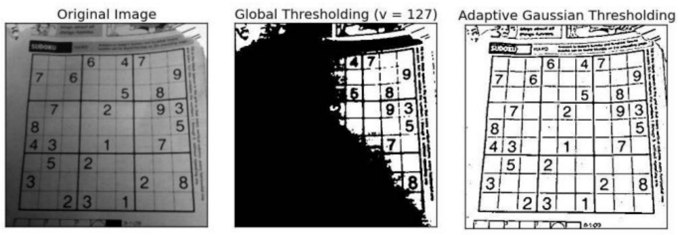
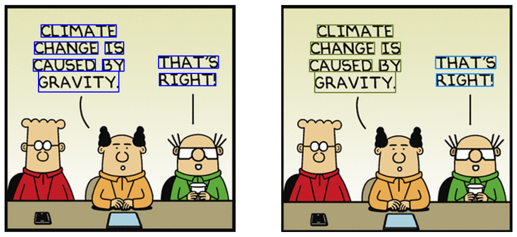


Fig. 4: Original img. vs. global threshold vs. adaptive threshold [23]

After binarization, contours are identified in the image. Only the outermost, top-level contours are interesting for this task, as those form the candidates for the panel bounding boxes (see Fig. 3c-d).

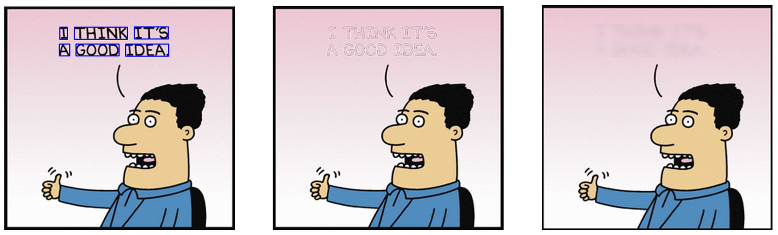
After identifying the outermost contours, they are filled in with white color, but the resulting image still has some noise, see for example the *"WWW.PHDCOMICS.COM"* text on the bottom of Fig. 3e. A morphological opening operation is applied to remove the noise. It is a combination of erosion and dilation operations, allowing for the removal of the small elements of the image while preserving the shape and size of the larger ones.

The resulting image, as presented in Fig. 3f has no more noise. Contours present in that image, determine the position of the final bounding box, see Fig. 3g. Once the algorithm identifies the positions of all the panel frames, the original image is divided into separate illustrations, by extracting the areas surrounded by the bounding boxes and saving them as new images.


CLIMATE CHANGE IS CAUSED BY GRAVITY
THAT'S RIGHT!
CLIMATE CHANGE IS CAUSED BY GRAVITY
THAT'S RIGHT!

(a) All b-boxes **(b)** Identified clusters

Fig. 5: Example of bounding-box clustering. Illustration taken from https://dilbert.com.



(a) Initial illustration **(b)** Before denoising **(c)** Final image

Fig. 6: Example of text removal from an illustration. Illustration taken from https://dilbert.com.

D. Text extraction

After the comic is segmented into individual illustrations, the text extraction has to be performed on each illustration. That can be achieved using optical character recognition — a conversion of image representation containing text into plain text strings.

1) Engines: There is a wide selection of OCR software on the market, therefore it is not feasible to test all of it, but two most popular engines are selected:

- *Tesseract* [24] - the leading open-source OCR engine.
- Google's *Vision API OCR* [25] - the state-of-the-art commercial OCR API.

2) Pre-processing: Pre-processing techniques, such as upscaling and binarization, can be applied to images to improve the performance of OCR [26].

a) Upscaling: Character height is considered a key factor for OCR output quality, and for optimal performance, it should be between 20-40 pixels. Unfortunately, the character heights in the comic datasets are often much smaller — an up-scaling step is needed. To know the re-scaling factor, one has to find the character height in the original image. For this purpose, an initial OCR pass is performed on the unprocessed image, and the character height is defined as the median of the heights of the bounding boxes returned by the engine. Then, images are re-scaled using cubic interpolation by a factor of $\frac{desired\ letter\ height}{initial\ letter\ height}$, where a common value for the desired letter height is 30 pixels.

b) Binarization: : The images are then binarized using adaptive thresholding [12], and fed into the final OCR pass, potentially resulting in better accuracy than before pre-processing.

3) Post-processing:

a) Clustering: The OCR output consists of detected words along with their bounding boxes, see Fig. 5a. The boxes are initially ordered top-to-bottom, left-to-right, as in a standard printed text page. However, this does not work for comics, as it does for example, for Fig. 5a, this would result in the following output:

"climate change is caused by that's gravity. right!"

The source of this issue is the lack of information about the comic bubbles composition, which is crucial for determining the correct order of words in comic dialogues. Comics are read bubble by bubble, rather than simply line by line. To correct the output, the bounding boxes need to be grouped into clusters corresponding to bubbles, as presented in Fig. 5b. We calculate the bubbles' centers and use them to sort bubbles by their centers' x and y positions. We can then read the text individually for each bubble, and concatenate the results to obtain the corrected transcription:

Bubble 1: "climate change is caused by gravity"
Bubble 2: "that's right!"
Concatenated: "climate change is caused by gravity. that's right!"

The bubble grouping can be performed using agglomerative hierarchical clustering [13, 27]. Initially, each bounding box starts alone, in a singleton cluster. Then clusters are merged until all the pairwise distances between clusters are higher than a certain threshold. To determine the distance between two clusters, a single linkage approach is used - the distance between clusters is the minimal distance between a pair consisting of elements of those two clusters (one from each). The distance between two bounding boxes is defined as the sum of minimum spacings between their edges in x and y directions, see Fig 7. Additionally, if there is overlap in an axis, the spacing in that axis is set to 0. The distance threshold for clusters can be determined based on the calculated letter height — spacing between two lines of text within one bubble would rarely be larger than the height of one letter.

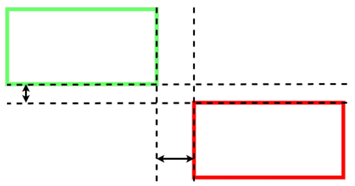


Fig. 7: x and y spacings between two bounding boxes.

b) Autocorrect: Single character mistakes are very common in the OCR output — often the majority of the characters are detected correctly, but some letters are classified as a different character than they actually are. In that cases, it can

TABLE I: Web scraping and panel extraction evaluation.

(a) Web scraping - average time to scrape one panel (in seconds), and speed-up factor achieved by the use of parallelization. Time estimates based on scraping 1000 *Dilbert*, 1000 *PHD Comics* and 1000 *Garfield* images from the web.

Result or Dataset	**Dilbert**	**PHD Comics**	**Garfield**	**Average**
1 thread	1.46	2.72	0.43	**1.54**
10 threads	0.22	0.18	0.046	**0.149**
speed-up	6.6	15.1	9.3	**10.33**

(b) Comparison of panel extraction performance between our method (DCP) and Kumiko [20]: single panel and full strip success rates, intersection-over-union scores, and time efficiency. Tested on 300 comic strips with total of 1118 panels.

Metric or dataset	**Dilbert**		**PHD Comics**		**Garfield**	
	Kumiko	DCP	Kumiko	DCP	Kumiko	DCP
Panel succ. rate	97%	100%	92%	99%	91%	91%
Strip succ. rate	95%	100%	78%	96%	73%	72%
Average IoU	0.99	0.99	0.96	0.98	0.97	0.95
Time per comic	680ms	2.3ms	400ms	1.1ms	890ms	1.2ms

be beneficial to make use of spelling correction algorithms. TextBlob [28] library provides an autocorrect implementation based on Peter Norvig's [29] correction algorithm. That implementation is used to correct the OCR output, aiming to reduce the error.

4) Text removal from image: Removing text from comic illustrations is an important feature for several use cases. For example, with the help of automated translation tools, one can remove the original text, and print new, translated text on the image. Moreover, the text is a prominent feature of the image, that will cause severe noise problems when trying to train models for illustration generation - removing it simplifies the data and allows the models to focus on the primary elements of the illustration, such as characters or objects.

The text removal method proposed in this paper is based on the bounding boxes found by the OCR model. For each bounding box, a binary mask is established based on binary thresholding, 1s correspond to dark spots (letters), and 0s correspond to the background. Then, the background color is calculated by taking the average color value of non-text pixels. The letter pixels are then colored with the background color, as presented in Fig. 6b. To give a smoother look to the image, denoising, and blurring are applied, see 6c.

IV. Experiments and Results

A. Datasets

The evaluation is performed by attempting to automatically construct illustration-transcription datasets for *PHD Comics*, *Garfield* and *Dilbert* — three popular comic series. All of them have been consistently updated with new comics for decades, providing thousands of comic strips to work with.

B. Comic scraping

Web scraper is evaluated by downloading comic strips from *dilbert.com* and *phdcomics.com* and *pt.jikos.cz/garfield*. Over 14000 *Garfield*, 12000 *Dilbert* and 2100 *PHD Comics* strips are downloaded. Multi-threaded scraping is significantly faster than single-threaded, with 10 speed-up factor for *Garfield*, 6.5 for *PHD Comics*, and 15.1 for *Dilbert*, see Table Ia. To give a better idea of the scale, scraping all 12000 *Dilbert* comics would take approximately 9 hours with a single thread, but only 35 minutes with 10 threads.

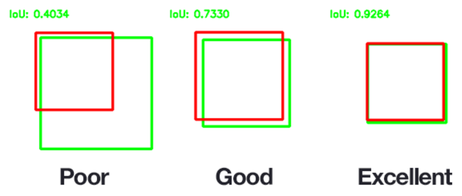


Fig. 8: Example IoU scores for bounding box evaluation. Taken from Wikipedia [30].

C. Panel extraction

When evaluating panel extraction, one needs to compare detected segments' locations with the manually marked, ground truth locations. Intersection over Union (IoU), also known as the Jaccard index, is a commonly used metric for measuring region overlap for all kinds of segmentation or detection tasks. The IoU is defined as the size of the intersection divided by the size of the union of two sets [30], see Equation 1.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

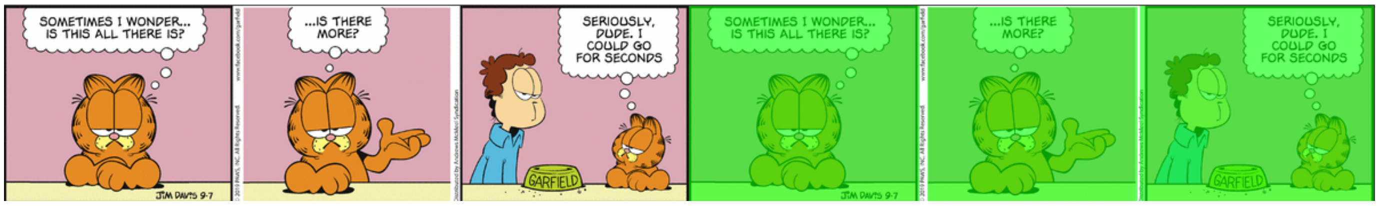
In the case of panel extraction, IoU is the area of the overlap between the detected bounding box and the ground-truth one, divided by the area of the union of those two bounding boxes. The IoU score will approach 1 if the bounding boxes are the same and 0 if there is no overlap between them, see Fig. 8. In this experiment, a panel is marked as correctly detected if the IoU for the detected and ground truth bounding boxes is at least 0.9. The comic strip is marked as correctly segmented if all panels are correctly detected.

Testing the panel extraction is done using comic strips from three different sources: *Dilbert*, *PHD Comics* and *Garfield*[1]. The test dataset consisted of 300 comics, containing 1118 panels. The results of the evaluation are presented in Table Ib.

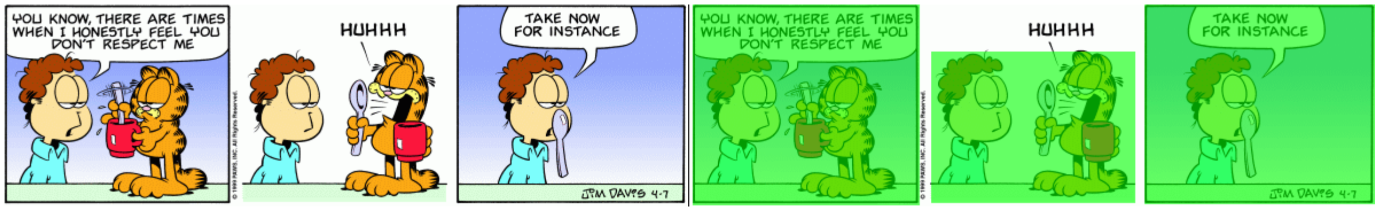
Overall, the proposed panel extraction algorithm achieves almost perfect results on *Dilbert* and *PHD comics* comics — leveraging the presence of the frames enables outperforming Kumiko. The performance on *Garfield* is noticeably worse, as no frames are present for some of the panels, making it harder to find the panel boundaries — see Fig. 9b for an example.

[1]Panel extraction for Garfield is performed using global threshold binarization rather than adaptive binarization, as it performed better when no frames are present.

Fig. 9: Examples of panel extraction results — the detected panels are represented by the green areas. (Comic strips from *Garfield* [1])


SOMETIMES I WONDER... IS THIS ALL THERE IS?
...IS THERE MORE?
SERIOUSLY, DUDE. I COULD GO FOR SECONDS
GARFIELD
JIM DAVIS 9-7
SOMETIMES I WONDER... IS THIS ALL THERE IS?
...IS THERE MORE?
SERIOUSLY, DUDE. I COULD GO FOR SECONDS
GARFIELD

(a) Correct segmentation example: all three panels are detected correctly.


YOU KNOW, THERE ARE TIMES WHEN I HONESTLY FEEL YOU DON'T RESPECT ME
HUHHH
TAKE NOW FOR INSTANCE
JIM DAVIS 4-7
YOU KNOW, THERE ARE TIMES WHEN I HONESTLY FEEL YOU DON'T RESPECT ME
HUHHH
TAKE NOW FOR INSTANCE

(b) Incorrect segmentation example: the middle panel is not detected correctly, as there is no clear border around it.

When it comes to efficiency, our algorithm is significantly faster than Kumiko, making it more suited for processing large datasets. It also has a significant advantage over deep-learning-based panel detection techniques - no dataset-specific training is needed, the method can be directly applied to any other comic.

D. Text extraction

Evaluation of text extraction is performed by comparing ground-truth, manual transcriptions of comic strips with the output of automated transcription using OCR. The evaluation is conducted on *Garfield*, *Dilbert* and *PHD Comics* datasets, containing 500, 500, and 100 annotated comic strips respectively. The *Garfield* and *Dilbert* transcriptions are available online in Alfred Arnold's transcription archive [31], and the *PHD Comics* annotations are obtained via manual transcriptions.

The Levenshtein distance, also known as the edit distance, is used as a primary metric for text extraction evaluation. Given two strings, the Levenshtein distance is defined as the minimal number of single-character edits (insertions, deletions, substitutions) needed to change one of the strings into the other:

$$L_{dist}(s_t, s_d) = C_{ins}(s_t, s_d) + C_{del}(s_t, s_d) + C_{sub}(s_t, s_d) \quad (2)$$

Where s_t is the ground-truth string, and s_d is the detected string. The distance can be normalized by dividing by the length of the ground truth string:

$$L_{dist\,norm}(s_t, s_d) = min(1, \frac{L_{dist}}{\mid s_t \mid}) \quad (3)$$

Given the comics text is almost always capitalized, the evaluation of the transcriptions is performed in a case-insensitive manner — the strings are converted to lowercase before comparison. Therefore, no distinctions are made between lowercase and uppercase letters; for example, *"CAT"* and *"cat"* are treated as the same string with distance 0.

To evaluate the impact of this paper's contribution to comic dialogue transcription, a baseline scenario is established: feeding the entire comic strip into the OCR engines, without pre-processing and dividing it into panels, denoted as *Exp. #1* in Table IIa. As presented in Table IIb, results achieved using the baseline approach are extremely inaccurate, making them completely unusable. Two primary reasons for the failure are:

1) The OCR picks up a lot of text from outside the actual illustrations. That text is not part of the dialogues, it mostly contains other information such as publication dates, comic artist's name, or website URLs.
2) As there is no information about panel division in this experiment, the OCR engines struggle with determining the correct order of the output — e.g. some text from the second panel can appear before some parts of text from the first panel.

The first major improvement to this scenario is experiment **#2** from Table IIa, where instead of scanning the whole image at once, the OCR is performed separately on each panel, and the results are then concatenated. We can observe a significant decrease in error rates. All the later experiments are conducted on separate panels, rather than on the full comic strip.

In the next two experiments — #3 and #4 from Table IIa — the impact of pre-processing techniques is evaluated. As presented in Table IIb - #3, adding an up-scaling step has a minor, but positive impact on the performance — especially in the case of *PHD Comics*, where the initial image resolution is low for some of the older strips. Experiment #4 results indicate that adding a binarization step has a slightly negative impact on the outcome, contrary to general OCR pre-processing recommendations from the literature. Based on these evaluation results, in the later experiments the binarization step is skipped, and only the up-scaling step is applied.

Experiment #5 aims to evaluate the impact of adding a

TABLE II: Text extraction evaluation - experiment on 500 Dilbert, 500 Garfield and 100 PHD Comics with ground-truth strings obtained via manual transcription.

(a) Experiment setup: six experiments are conducted to evaluate text extraction. Experiments test OCR on full and segmented strips, using the proposed pre-processing and post-processing techniques.

Exp. no.	Segmentation	Pre-processing		Post-processing	
		re-sizing	binarization	clustering	autocorrect
Exp. #1	✗	✗	✗	✗	✗
Exp. #2	✓	✗	✗	✗	✗
Exp. #3	✓	✓	✗	✗	✗
Exp. #4	✓	✓	✓	✗	✗
Exp. #5	✓	✓	✗	✓	✗
Exp. #6	✓	✓	✗	✓	✓

(b) Experiment results: normalized Levehnstein distance between detected and ground-truth transcriptions. Comparison of Vision API and Tesseract OCR on *Dilbert*, *Garfeld*, and *PHD Comics* datasets.

Exp. no.	Dilbert		PHD Comics		Garfield	
	Tess.	V. API	Tess.	V. API	Tess.	V. API
Exp. #1	0.68	0.61	0.731	0.538	0.650	0.381
Exp. #2	0.233	0.044	0.786	0.109	0.532	0.163
Exp. #3	0.222	0.044	0.698	0.104	0.501	0.159
Exp. #4	0.242	0.048	0.727	0.112	0.534	0.150
Exp. #5	**0.188**	**0.032**	0.699	**0.075**	**0.468**	**0.120**
Exp. #6	0.276	0.097	**0.694**	0.0781	0.485	0.121

bounding box clustering step on the OCR performance. Table IIb shows a significant positive impact on the accuracy — clustering reduces the error rates by up to 30%. This shows, that a significant fraction of the errors is caused by the wrong ordering of the output words due to a lack of information about the comic speech bubbles. Clustering fixes that issue for most data points.

Finally, experiment #6, from Table IIa evaluates the impact of auto-correcting the OCR output on the extraction error. Intuitively, one could expect some improvement from dictionary-based correction, but the results in Table IIb show an opposite effect. One explanation for this could be that comics contain a lot of names, onomatopoeias, and exclamations — such as *"Dilbert"*, *"Woo"* or *"Pow"* — which are not present in the dictionary and get mistakenly corrected into other words.

Overall, the final performance of the OCR is satisfactory, but not perfect. Vision API performs better than Tesseract in all cases. Tesseract completely fails with *PHD Comics* and *Garfield*, in a big part of the comics it does not detect text at all. Panel separation and clustering have a significant, positive impact on the performance, but the other elements of the proposed method do not bring improvement. Best error rates of 0.03 on *Dilbert*, 0.07 on *PHD Comics*, and 0.12 on *Garfield* give a solid base for automatic transcription, but in the current state most likely the transcription would still have to be corrected by a human.

V. Discussion, conclusions and future work

The purpose of this paper was to design, implement, and evaluate an automated dataset construction pipeline for building an illustration-transcription comics dataset. To do so, web-scraping was applied to automatically download the comic strips, image processing techniques were utilized to divide the comics into individual illustrations, and OCR was used to automatically generate transcriptions.

The web scraping technique proved successful in the experiments. We were able to download thousands of *Garfield*, *Dilbert* and *PHD Comics* strips, and thanks to the multithreaded implementation, we achieved an average rate of between 5 and 20 comics per second.

The proposed panel extraction method achieved success rates for *Dilbert* and *PHD Comics*, achieving 100% and 97% success rates respectively, outperforming the baseline algorithm in terms of both accuracy and efficiency. Unfortunately, it failed to detect a significant fraction of *Garfield* panels, achieving only a 71% success rate. The source of the errors was the lack of clear panel boundaries in some of the strips. The proposed algorithm used contour detection and thresholding to detect the panel boundary, therefore it dealt flawlessly with panels that had frames, but struggled when no frame was present.

The automatic text transcription was the most challenging stage of the process, as the existing OCR solutions performed poorly in their out-of-the-box state. Moreover, the proposed OCR pre-processing methods, such as binarization and up-scaling, brought no significant improvement to the performance. However, performing OCR on individual illustrations and correcting the order of the output by grouping the text bounding boxes into speech bubbles reduced the error rates by a factor of 7. It was thought that performing dictionary autocorrect on the OCR output would bring further improvement, but the effect was the opposite, possibly due to the presence of non-dictionary words, like onomatopeias and exclamations in the comic dialogues. Overall, the final OCR output is fairly close to the true transcriptions, with normalized Levenshtein distances of 0.03, 0.07, and 0.12 for *Dilbert*, *PHD Comics*, and *Garfield* respectively.

In general, the proposed pipeline can successfully construct a dataset of comic illustrations and transcriptions for most data points, but the output still contains an observable amount of errors. Part of the errors can be attributed to the mistakes in segmenting the comic strips where no clear panel frames are present. It could be beneficial to conduct further research to develop a solution that can deal with such cases. The majority of the errors occur at the text extraction stage - there might be a need to construct a human-in-the-loop software, including a tool for manual correction of the output transcriptions. Another possibility for improvement is to experiment with dataset-specific training of the OCR models on a small, manually annotated dataset. Finally, some text-region detection algorithms, such as the EAST text detector [32], could be used to detect candidate text areas and feed those into the OCR pipeline instead of the whole comic strips, potentially resulting in better OCR accuracy.

VI. Responsible research

The following three paragraphs reflect on the ethical and legal implications of this research project, and the reproducibility of results achieved in the experiments.

A. Copyright issues

The data used for experiments was obtained via web scraping, which is a topic of debates concerning legal issues such as copyright and privacy violations [33]. There is no risk of privacy violations in the case of this research project — all of the data points are publicly available artworks, created to be shared with a wide audience. However, the copyright violation is a real threat: the comic strips from *PHD Comics* and *Dilbert* are the intellectual property of their creators, and cannot be freely distributed by a third party. Therefore, to avoid any concerns regarding copyright violation, the datasets used for experiments will not be published. Some of the *PHD Comics* strips are still used in the paper to illustrate the methods and experiments, but this kind of usage is explicitly listed as permitted on *PHD Comics* website [34].

B. Potential software misuse

Another possible ramification of this project is the potential misuse of the published software. For instance, the proposed scraping mechanism could be used to mass-download copyrighted comics, which could then be republished illegally on a third-party website. Moreover, the text removal method implemented in the project could be used to clear the existing dialogues from a comic and add new ones, creating an alternative story. This usually goes against the comic publisher's regulations, as it involves producing derivative work from copyrighted content. In general, software misuse cannot be fully prevented, but explicit warning regarding this topic is included in the software documentation.

C. Reproducibility

Reproducibility of results is a crucial aspect of research, but, unfortunately, it is often overlooked by scientists [35], also in the computer science field [36]. To ensure the reproducibility of the results achieved in this research, the source code will be published as open-source software on github.com [2], along with a usage guide. This way, the experiments mentioned in the paper can be easily repeated by interested parties and compared with new research. Moreover, the code can also be forked and modified to be used in a different context or improved by new ideas.

[2]Project repository: https://github.com/mstyczen/comic-dcp

References

[1] *Garfield*. URL: https://garfield.com/.

[2] *PHD Comics*. URL: http://phdcomics.com/.

[3] *Dilbert*. URL: https://dilbert.com/.

[4] Faizan Ahmad, Aaima Najam, and Zeeshan Ahmed. "Image-based face detection and recognition:" state of the art"". In: *arXiv preprint arXiv:1302.6379* (2013).

[5] Ian J Goodfellow et al. "Generative adversarial networks". In: *arXiv preprint arXiv:1406.2661* (2014).

[6] Han Zhang et al. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5907–5915.

[7] Scott Reed et al. "Generative Adversarial Text to Image Synthesis". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 1060–1069. URL: http://proceedings.mlr.press/v48/reed16.html.

[8] Rada Mihalcea, Carlo Strapparava, and Stephen Pulman. "Computational models for incongruity detection in humour". In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer. 2010, pp. 364–374.

[9] Yuji Roh, Geon Heo, and Steven Euijong Whang. "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective". In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2021), pp. 1328–1347. DOI: 10.1109/TKDE.2019.2946162.

[10] Hillary Sanders and Joshua Saxe. "Garbage in, garbage out: How purportedly great ML models can be screwed up by bad data". In: *Proceedings of Blackhat 2017* (2017).

[11] Cem Dilmegani. *Current State of OCR: Is it a solved problem in 2021?* 2021. URL: https://research.aimultiple.com/ocr-technology/#:~:text=of%20OCR%20tools?-,OCR%20is%20not%20a%20stand-alone%20solution%20in%20human-machine,structured%20data%20from%20their%20documents..

[12] Jamileh Yousefi. "Image binarization using Otsu thresholding algorithm". In: *Ontario, Canada: University of Guelph* (2011).

[13] William HE Day and Herbert Edelsbrunner. "Efficient algorithms for agglomerative hierarchical clustering methods". In: *Journal of classification* 1.1 (1984), pp. 7–24.

[14] H. Ng and S. Winkler. "A data-driven approach to cleaning large face datasets". In: *2014 IEEE International Conference on Image Processing (ICIP)*. 2014, pp. 343–347. DOI: 10.1109/ICIP.2014.7025068.

[15] Ryan Mitchell. *Web scraping with Python: Collecting more data from the modern web.* " O'Reilly Media, Inc.", 2018.

[16] Van Nguyen Nhu, Christophe Rigaud, and Jean-Christophe Burie. “What do We Expect from Comic Panel Extraction?” In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Vol. 1. 2019, pp. 44–49. DOI: 10.1109/ICDARW.2019.00013.

[17] Toru Ogawa et al. “Object detection for comics using manga109 annotations”. In: *arXiv preprint arXiv:1803.08670* (2018).

[18] Anh Khoi Ngo Ho, Jean-Christophe Burie, and Jean-Marc Ogier. “Panel and speech balloon extraction from comic books”. In: *2012 10th IAPR International Workshop on Document Analysis Systems*. IEEE. 2012, pp. 424–428.

[19] Xufang Pang et al. “A Robust Panel Extraction Method for Manga”. In: *Proceedings of the 22nd ACM International Conference on Multimedia*. MM ’14. Orlando, Florida, USA: Association for Computing Machinery, 2014, pp. 1125–1128. ISBN: 9781450330633. DOI: 10.1145/2647868.2654990. URL: https://doi.org/10.1145/2647868.2654990.

[20] *Kumiko*. URL: https://github.com/njean42/kumiko/.

[21] Christophe Rigaud et al. “Toward speech text recognition for comic books”. In: *Proceedings of the 1st International Workshop on coMics ANalysis, Processing and Understanding*. 2016, pp. 1–6.

[22] Christophe Ponsard, Ravi Ramdoyal, and Daniel Dziamski. “An OCR-Enabled Digital Comic Books Viewer”. In: *Computers Helping People with Special Needs*. Ed. by Klaus Miesenberger et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 471–478. ISBN: 978-3-642-31522-0.

[23] *Image Thresholding*. URL: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html.

[24] *Tesseract*. URL: https://github.com/tesseract-ocr/tesseract.

[25] *Google Vision API*. URL: https://cloud.google.com/vision/docs/ocr.

[26] Wojciech Bieniecki, Szymon Grabowski, and Wojciech Rozenberg. “Image Preprocessing for Improving OCR Accuracy”. In: *2007 International Conference on Perspective Technologies and Methods in MEMS Design*. 2007, pp. 75–80. DOI: 10.1109/MEMSTECH.2007.4283429.

[27] *Agglomerative clustering - scikit-learn documentation*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[28] *TextBlob: Simplified Text Processing*. URL: https://textblob.readthedocs.io/en/dev/index.html.

[29] Peter Norvig. *How to Write a Spelling Corrector?* URL: http://norvig.com/spell-correct.html.

[30] *Jaccard index*. May 2021. URL: https://en.wikipedia.org/wiki/Jaccard_index.

[31] *Alfred Arnold’s FTP server with comic transcriptions*. URL: http://john.ccac.rwth-aachen.de:8000/ftp/dilbert/.

[32] Xinyu Zhou et al. “East: an efficient and accurate scene text detector”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 5551–5560.

[33] Vlad Krotov and Leiser Silva. “Legality and ethics of web scraping”. In: (2018).

[34] *PHD Comics: About*. URL: https://phdcomics.com/about.php.

[35] Monya Baker. “Reproducibility crisis”. In: *Nature* 533.26 (2016), pp. 353–66.

[36] Matthew Hutson. *Artificial intelligence faces reproducibility crisis*. 2018.