Suitability of Optical Character Recognition (OCR) for Multi-domain Model Management

Weslley Torres, Mark G. J. van den Brand, and Alexander Serebrenik

Eindhoven University of Technology, Eindhoven, The Netherlands
{w.silva.torres,m.g.j.v.d.brand,a.serebrenik}@tue.nl

Abstract. The development of systems following model-driven engineering can include models from different domains. For example, to develop a mechatronic component one might need to combine expertise about mechanics, electronics, and software. Although these models belong to different domains, the changes in one model can affect other models causing inconsistencies in the entire system. There are, however, a limited amount of tools that support management of models from different domains. These models are created using different modeling notations and it is not plausible to use a multitude of parsers geared towards each and every modeling notation. Therefore, to ensure maintenance of multi-domain systems, we need a uniform approach that would be independent from the peculiarities of the notation. Meaning that such a uniform approach can only be based on something which is present in all those models, i.e., text, boxes, and lines. In this study we investigate the suitability of optical character recognition (OCR) as a basis for such a uniformed approach. We select graphical models from various domains that typically combine textual and graphical elements, and we focus on text-recognition without looking for additional shapes. We analyzed the performance of Google Cloud Vision and Microsoft Cognitive Services, two off-the-shelf OCR services. Google Cloud Vision performed better than Microsoft Cognitive Services being able to detect text of 70% of model elements. Errors made by Google Cloud Vision are due to absence of support for text common in engineering formulas, e.g., Greek letters, equations, and subscripts, as well as text typeset on multiple lines. We believe that once these shortcomings are addressed, OCR can become a crucial technology supporting multi-domain model management.

Keywords: Model Management · Systems Engineering · OCR.

1 Introduction

Model-driven engineering (MDE) has been used in diverse engineering fields such as software engineering [31], robotics [46], and automotive [42]. The promised benefits of using this approach include increased development speed, earlier system analysis, and more manageable complexity [45]. However, managing interrelated models of different domains is challenging [33].

Since models from different domains use different modeling languages, to manage interrelated models of different domains one has to use a technology that can handle these models independently of the language used. Graphical models from various domains typically combine textual and graphical elements such as boxes, lines, and arrows. Such models can be designed using different tools, and usually these tools can export the model in a structured format such as XML, or as an image format such as PNG or JPEG [37]. The ideal setting for data extraction would be if all models were available in a structured default format. However, the situation becomes more complex when only the image of the model is available, either because the code that generates the model is lost, the model is only available in a paper instead of a digital format, or simply because the modeling tool does not export the model in the desired format.

Therefore, to ensure maintenance of multi-domain systems, we need a uniform approach that would be independent from the peculiarities of the notation. This also means that such a uniform approach can only be based on something which is present in all those models, i.e., text, boxes, and lines.

We believe that optical character recognition (OCR) can become a part of the uniformed approach. OCR is a collection of techniques aiming at recognizing text from handwritten or printed document and exporting the result as a machine-encoded text. Originally developed to support blind people [34], OCR is used nowadays for instance for automatic number plate recognition, and passport data recognition in airports [40].

Hence, in the current work we evaluate the precision of OCR on models from different domains focusing on text-recognition without looking for additional shapes. The aim is to answer the following research questions:

- **RQ1)** How accurate are off-the-shelf OCR services for extracting text from graphical models?
- **RQ2)** What are the common errors made by OCR services on models from different domains?

To answer RQ1 we apply Google Cloud Vision[1] and Microsoft Cognitive Services[2] to a collection of 43 models from different domains. We observe that Google Cloud Vision outperforms Microsoft Cognitive Services, being able to detect 70% of textual elements as opposed to 30% by Microsoft. To answer RQ2 we inspected errors made by Google Cloud Vision. We observed that in 100% of the cases, Google Cloud Vision was not capable of identifying Greek letters, subscripts, and text typeset in multiple lines. The lack of support for these elements represents the main challenge for adoption of OCR in multi-domain model management.

To encourage replication of our work the data we have collected and the source code we have used to perform the analysis has been made available on: bit.ly/ShareDataOCR

[1] https://cloud.google.com/vision/
[2] https://azure.microsoft.com/en-us/services/cognitive-services/

The results presented in this work can be used to guide further research on OCR for multi-domain model management. For future work, we plan to combine different kind of image processing techniques to improve the results, for instance, being able to detect formulas, and semantic information by analyzing boxes, lines, and arrows.

2 Methodology

To answer RQ1 we apply Google Cloud Vision and Microsoft Cognitive Services to a collection of 43 models from different domains. To answer RQ2 we focus on the OCR service that has been shown to perform better on RQ1 and inspect the errors made by the service.

2.1 Models Selection

For reproducibility reasons, we arbitrarily select models from two UML open repositories [1, 29], three control system engineering papers [30, 36, 47], and the example catalog of MatLab Simulink[3]. In total we analyzed 43 models as presented in Table 1. We only require the models to be graphical models, i.e., they must contain a mix of textual and graphical elements.

We select MatLab Simulink models because of its high adoption by the industry. These models are available on the official website as example catalog and they are used to describe control systems from different domains including automatic climate control, robot arm control, and fault-tolerant fuel control. We also include models from three scientific papers on control system engineering. The models from these papers are an intelligent control architecture of a small-Scale unmanned helicopter, an actuator control system, and a x-ray machine.

Among the UML models we focus on Class Diagram, Sequence Diagram, and Use Case Diagrams. These models are stored in two repositories: Git UML [1] and Models-db [29]. The former automatically generates diagrams from source code stored in git repositories. Models-db is automatically populated by crawlers identifying models from public GitHub repositories.

Source	Models	#Models
Ai et al. [30]	Figures 1, 4, 5	5
Kaliappan et al. [36]	Figures 1-3, 5, 7	5
Tovar-Arriaga et al. [47]	Figures 1, 5–8, 10, 15	7
UML	[12–28]	17
MatLab Simulink	[3–11]	9
Total		43

Table 1. List of models used to answer RQ1.

[3] https://www.mathworks.com/products/simulink.html

2.2 Text Extraction

In order not to bias the evaluation towards a specific engineering domain, we opt for general-purpose OCR techniques.

Several OCR serves are available off the shelf, including Google Cloud Vision, Microsoft Cognitive Services, and Amazon AWS Rekognition[4]. For this work, we select the Google Cloud Vision and Microsoft Cognitive Services: these services have been shown to be effective in recognizing text from the photos of the pages of the Bible [44], and to outperform Amazon AWS on images of business names or movie names [2].

2.3 Measures for Accuracy

The validation consists of manually identifying the text from graphical models, and comparing the text extracted by OCR to the manually identified text. When deciding whether the OCR-extracted text matches the manually extracted one we do not distinguish between the letter case, i.e., *Velocity* is seen as the same as *veLoCitY*. We do distinguish between differently chunked texts, i.e., given the manually identified text *Velocity control* an OCR service extraction of *Velocity* and *Control* as two separate texts will be seen as wrong.

As common in information retrieval tasks we report precision, recall, and F-measure, i.e., the harmonic mean of precision and recall. In our context *precision* is the fraction of OCR-extracted texts that are also manually extracted compared to all OCR-extracted texts, and *recall* is the fraction of OCR-extracted texts that are also manually extracted compared to all manually extracted texts.

3 Results

3.1 R1: How accurate are off-the-shelf OCR services for extracting text from graphical models?

Overview In overall, Google Cloud Vision correctly detected 854 out of 1,232 elements, while Microsoft Cognitive Services correctly detected 388 elements. This observation concurs with previous evaluations of these OCR services. Indeed, on the photos of the pages of the Bible Reis et al. [44] observed that Google Cloud Vision had a relative effectiveness of 86.5% as opposed to 77.4% of Microsoft Cognitive Services. On images of business names or movie names [2] Google Cloud Vision achieved 80% of both precision and recall as opposed to 65% of precision and 44% of recall of Microsoft Cognitive Services.

Hence, we hypothesize that also on our dataset Google Cloud Vision will outperform Microsoft Cognitive Services in terms of both precision and recall. Formally, we state the following hypotheses:

- H_0^p: The median difference between the *precision* for Google Cloud Vision and Microsoft Cognitive Services is zero.

[4] https://aws.amazon.com/rekognition/

- H_a^p: The median difference between the *precision* for Google Cloud Vision and Microsoft Cognitive Services is greater than zero.
- H_0^r: The median difference between the *recall* for Google Cloud Vision and Microsoft Cognitive Services is zero.
- H_a^r: The median difference between the *recall* for Google Cloud Vision and Microsoft Cognitive Services is greater than zero.

To test these hypotheses we perform two paired Wilcoxon signed-rank tests, one for precision and another one for recall. The p-values obtained for precision and recall are 1.9×10^{-7} and 2.8×10^{-9}, respectively. Hence, we can reject H_0^p and H_0^r and state that Google Cloud Vision outperforms Microsoft Cognitive Services.

To illustrate this argument consider Figure 1. It summarizes precision (y-axis) and recall (x-axis) organized by the type of models. Indeed, we can observe that while precision and recall obtained by Google Cloud Vision *mostly* exceed 0.5, precision and recall obtained by Microsoft Cognitive Services are *mostly* below 0.5. Moreover, the data for both Google Cloud Vision and Microsoft Cognitive Services suggests a linear relation between precision and recall: indeed, while the *number* of textual elements extracted by OCR tools is often close to the number of manually identified textual elements, the textual elements themselves are imprecise.

Finally, while Google Cloud Vision extracted *some* textual information from all models, Microsoft Cognitive Services failed on two models: Matlab Simulink model [6] and Figure 4.b from the paper by Ai et al. [30].

Performance on Models of Different Domains While the previous discussion indicates that overall Google Cloud Vision outperforms Microsoft Cognitive Services, *a priori* this does not imply that this should also be the case for models of different domains. This is why we formulate the corresponding hypotheses separately for UML diagrams, Matlab Simulink models, and models from scientific papers. We test these hypotheses using paired Wilcoxon signed-rank tests, one for precision and another one for recall. However, since we perform multiple comparisons, we need to adjust the p-values to control for the false discovery rate. We use the method proposed by Benjamini and Hochberg [32].

The adjusted p-values are below the commonly used threshold of 0.05 for five of the six comparisons (three types of models $\times$ precision or recall). We conclude that Google Cloud Vision outperforms Microsoft Cognitive Services: for models of all domains in terms of recall; for UML diagrams and Matlab Simulink models in terms of precision as presented in Table 3.1.

Performance on Individual Models Figure 2 shows that the F-measure for Google Cloud Vision is higher than for Microsoft Cognitive Services on 33 models, as opposed to five models where Microsoft Cognitive Services scores higher. For the remaining five models the F-measures are equal.

Inspecting Figure 2 we also notice that for six models Microsoft Cognitive Services have precision equal to zero, i.e., either no textual elements have been

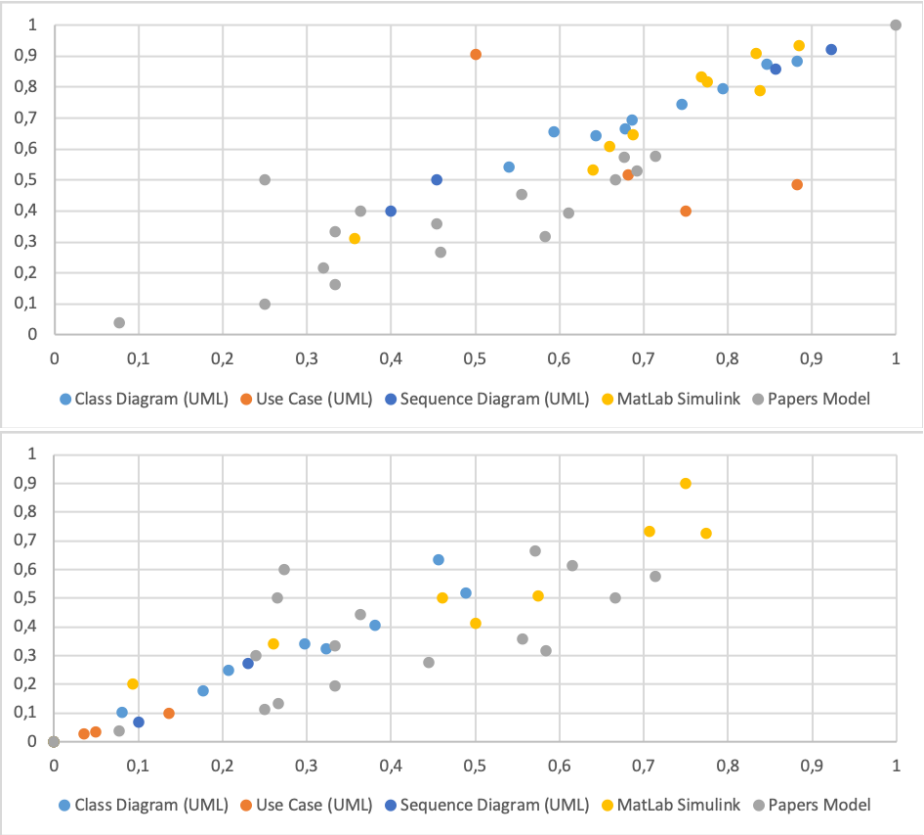

Fig. 1. Precision (y-axis) and recall (x-axis) obtained by Google Cloud Vision (top) and Microsoft Cognitive Services (bottom).

extracted (Matlab Simulink 4 and Paper model 3.3) or all textual elements extracted are wrong (UML Class Diagram 8, UML Use Case 4, UML Sequece Diagram 2 and 4). Unfortunately, we cannot precisely state the reasons why Microsoft Cognitive Services failed in process these models. Possible reasons could be related to the quality of the images, and size of font. However, these are unlikely to be the reasons for this fail, since all used images are in good quality and Google Cloud Vision managed to process the same images. In this study we did not look further in investing the reasons for the bad performance of Microsoft Cognitive Services.

Take away message

Google Cloud Vision was capable of detecting 70% of all elements, consistently outperforming Microsoft Cognitive Services.

Models	Precision	Recall
UML	Google Cloud Vision	Google Cloud Vision
Matlab Simulink	Google Cloud Vision	Google Cloud Vision
Scientific papers	-	Google Cloud Vision

Table 2. OCR service that presents statistically better results organized by the domain. The "-" means inconclusive result.

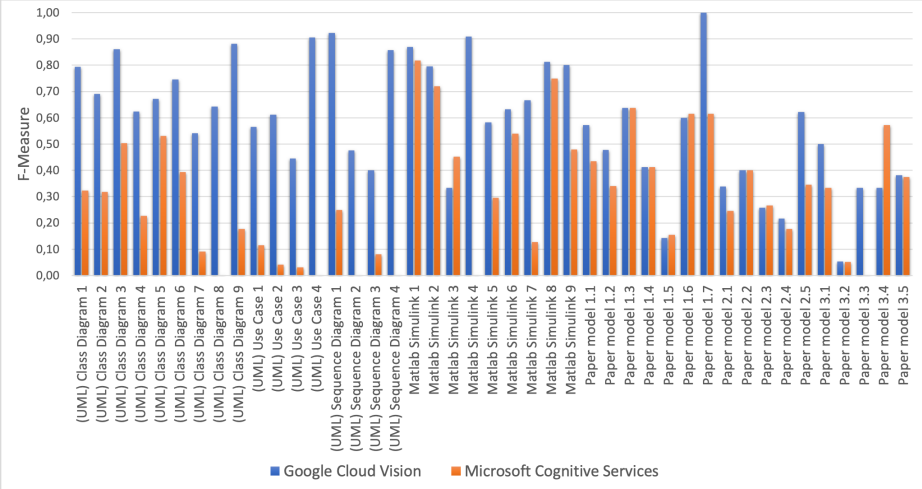

Fig. 2. F-Measure for all analyzed models

3.2 RQ2: What are the common errors made by OCR services on models from different domains?

Based on our answer to RQ1, one would prefer Google Cloud Vision as the OCR service to be integrated in a multi-domain model management solution. In this section we take a closer look at the errors made by Google Cloud Vision: addressing these errors is necessary in order to make OCR suited for multi-domain model management.

Table 3 summarizes the results of manual analysis of the errors made by Google Cloud Vision:

- The first category of errors is related to *non-alphanumerical characters* used in the models such as [, {, <, or _. These characters are sometimes confused with each other or missed by the OCR, e.g., the name of the element is *'file_version'* and OCR detects *'file version'*, without the underscore.
- Engineering models can involve *mathematical formulas* such as equations, including subscripts and Greek letters.
- The next group of errors is related to *spacing* and relative positioning of the textual elements. For example, due to space limitations text can be positioned on multiple lines, making OCR to misinterpret as one textual

Problem	Total	UML			MatLab	Paper
		CD	UC	SD		
Non-alphanumeric characters						
Brackets	1	0	0	0	1	0
Curly Brackets	2	0	0	2	0	0
Greater/Less Symbol	1	0	1	0	0	0
Parentheses	5	3	0	1	1	0
Slash	1	0	0	1	0	0
Underscore	8	7	0	0	1	0
Total	*18*	*10*	*1*	*4*	*3*	*0*
Mathematical formulas						
Equation	2	0	0	0	2	0
Subscript	2	0	0	0	0	2
Greek Letter	2	0	0	0	0	2
Total	*6*	*0*	*0*	*0*	*2*	*4*
Spacing						
Empty Space between Letters	15	6	4	3	1	1
Mix of Elements	8	4	0	0	3	1
Multi-line Text	28	4	3	0	7	14
Split Element	1	1	0	0	0	0
Total	*51*	*15*	*7*	*3*	*11*	*15*
Character confusion						
Character Confusion	8	1	2	0	3	2
Extra Char	11	4	2	0	2	3
Missing Char	14	5	1	0	3	5
Wrong Char	13	3	0	2	5	3
Total	*46*	*13*	*5*	*2*	*13*	*13*

Table 3. Number of models affected by the identified problems. CD - Class Diagram, UC - User Case, SD - Sequence Diagram

element but as two separate elements, we call this error as *Multi-line Text*. When this misinterpretation happens in a textual element positioned in one single line, we call this error as *Split Element*. The difference between *Multi-line Text* and *Split Element* is that the latter occurs on textual element is written in one single line but have an empty space between the words, causing this misinterpretation. The opposite of the error *Split Element* is *Mix of Elements*. *Mix of Elements* occurs when OCR mixes the name of different elements due to their proximity.

– Finally, the last group of errors is related to single-character errors such as characters being wrongly added, removed, or recognized. An example of such error is *Character Confusion*. This error occurs when OCR is not capable of identifying the letter due to the similarity to other letters. For instance, the name of the element is *'DeleteNodeById()'*. However, OCR interprets the capital letter 'i' as the lowercase 'l', returning *'DeleteNodeByld()'*. The difference between *Character Confusion* and *Wrong Char* is that the former

Problem	#Affected Models	#Candidate Models	#Affected Elements	#Candidate Elements
Non-alphanumeric characters				
Brackets	1	1	1	5
Curly Brackets	2	2	8	9
Greater/Less Symbol	1	12	4	60
Parentheses	5	5	11	137
Slash	1	11	1	39
Underscore	8	8	59	156
Mathematical formulas				
Equations	2	2	2	2
Subscript	2	2	15	15
Greek letters	2	2	2	2
Spacing				
Multi-line Text	27	27	130	130
Split Element	1	39	2	334

Table 4. Candidate Elements are the elements that contain characters that can cause a problem. Candidate Models are the models that have the candidate elements.

occurs between similar characters, e.g., the letter 'o' and the number '0'. And *Wrong Char* occurs between any character.

Table 3 shows that errors present in the largest number of models are *Multi-line Text*, *Empty Space between Letters*, *Missing Char*, and *Wrong Char*. However, the number of models affected by the errors should be compared to the number of models that *can* be affected by those errors: while wrong characters might appear in any model, errors related to underscores can only be present if the models contain underscores.

Hence, Table 4 summarizes the number of models that can be affected (candidate models) and the models that are affected by errors. Similarly, it includes the number of elements that can be affected (candidate elements) and are affected by the errors.

Inspecting Table 4 we observe that the *Curly Brackets*, *Equations*, *Greek letters*, *Multi-line String*, *Parentheses*, *Subscript*, and *Underscore* occur in every single model that has the corresponding elements.

Even though *Parentheses*, and *Underscore* problems arise in 100% of the candidate models, Google Cloud Vision correctly identified 92% of textual elements that have parentheses and 60% of the textual elements that have underscores and this in sharp contrast with *Equations*, *Greek Letters*, *Multi-line String*, and *Subscript* that could not be recognized by Google Cloud Vision.

Take away message

The main OCR challenges are text that contains *Equations*, *Greek Letters*, *Multi-line String*, and *Subscript* due to the lower precision on correctly identify these elements.

4 Threats to Validity

As any empirical study our work is subject to threats to validity. Wohlin et al. [49] provide a list of possible threats that researchers can face during a scientific research. In this section, we describe the actions we took in order to increase the validity and decrease the threats.

Internal validity, concerns the unknown influences of independent variables can have on studies. In order to mitigate this concern, we have selected OCR services that have been evaluated by previous studies on different text recognition tasks. While the manual extraction of textual elements has been performed by one author only, the task is simple for an engineer and is unlikely to be affected by the subjectivity of their judgment.

External validity concerns the generalizability of the results and findings of the study. In order to mitigate this concern, we have diversified the collection of models analyzed to include models from different domains and different sources.

Construct validity concerns the issues related to the design of the experiment. In order to address this issue, we used metrics that were sufficiently defined in previous studies. Example of such metrics are precision, recall, and F-measure. We used these metrics to indicate which OCR service presents better performance.

Conclusion validity concerns about the relations between the conclusions that we draw and the analyzed data. In order to mitigate this concern, we paid special attention to use appropriate statistical techniques, and we described all decisions we made. Thus, this study can be replicated by other researchers, and we expect our results to be quite robust.

5 Discussion and Future Work

The results described in this paper can serve as a starting point for future research on the use of OCR for multi-domain model management, as well as for design of tools supporting multi-domain model management. We started by investigating accuracy of the off-the-shelf OCR services for extracting text from graphical models. Concurrent with the previous studies [2, 44] Google Cloud Vision outperformed Microsoft Cognitive Services on both precision and recall. However, the precision and recall values of Google Cloud Vision were not as high as the ones presented in the previous studies [2]. We believe this is due to the difference between the analyzed items: graphical models vs. business names.

As opposed to business names, graphical models often include mathematical elements such as Greek letters and subscripts, and non-alphanumeric characters. Moreover, extracting text from models that do not follow the same design rules, incurs additional challenges. Indeed, the precision and recall scores for models from scientific papers are much more spread out in Figure 1, than for models from other data sources.

Next, we investigated the common errors produced by Google Cloud Vision. We identified 17 different types of errors organized by four categories: *non-alphanumeric characters*, *mathematical formulas*, *spacing*, and *character confusion*. Most common errors are related to Spacing and Character confusion; however, the main challenges seem to be related to the mathematical formulas—not a single Greek letter, subscript or equation appearing in the models could be correctly identified.

As future work, we intend to focus on the main challenges we identified in Section 3.2. Furthermore, we want to evaluate different OCR techniques on additional kinds of graphical models, including, for instance, models drawn on whiteboards and hand-written models. Simultaneously, we intend to combine OCR with image processing to analyze graphical elements such as boxes, lines, and arrows presented in the models.

6 Related Work

To the best of our knowledge, there are no studies on the use of off-the-shelf OCR services on models from different domains. However, OCR has been applied to domain-specific models. Img2UML [37, 38] extracts UML Class Diagrams from images, identifying, e.g., class names, fields and methods. Img2UML uses Microsoft Office Document Imaging as the OCR technique for text recognition. While Img2UML is geared towards and evaluated on a specific domain, the techniques we have analyzed have been applied to models of multiple domains. Several studies have used OCR as part of a tool classifying images as UML diagrams: targeting class diagrams [35, 41], sequence diagrams [43] and component diagrams [41].

Going beyond engineering models, Reis [44] compare Google Cloud Vision and Microsoft Cognitive Services in recognizing text from the photos of the pages of the Bible. Additional comparison studies have been published by Mello and Dueire Lins [39] and Vijayarani and Sakila [48].

7 Conclusion

We presented a study of suitability of the off-the-shelf OCR services in the context of multi-domain model management. We evaluated performance of two well-known services, Google Cloud Vision and Microsoft Cognitive Services, on a collection of 43 models from different domains: 17 UML diagrams, 9 MatLab Simulink models and 17 models from scientific papers from the control system engineering domain.

We observed that Google Cloud Vision overall outperforms Microsoft Cognitive Services both in terms of precision and in terms of recall. This observation is consistent both with the previous work [2, 44] and with a follow-up study investigating performance of the two OCR-services on models of different domains.

Focusing on Google Cloud Vision, we identified a list of 17 kinds of errors distributed over four categories: non-alphanumeric characters, mathematical formulas, spacing and character confusion. Among these errors, the most common are related to text written on multiple lines, wrong/missing characters, and an empty space between letters. It is also important that in presence of multiline texts, Greek letters, subscripts, and equations because Google Cloud Vision failed every single time.

To conclude, we observe that even though Google Cloud Vision has some limitations, it produces satisfactory results. We believe that once the most problematic cases are solved, OCR can become a crucial technology to support multi-domain model management.

References

1. Git uml repository. https://www.gituml.com. Accessed: 2020-01-23
2. Image text recognition apis showdown. https://dataturks.com/blog/compare-image-text-recognition-apis.php. Accessed: 2020-01-08
3. Matlab simulink model 1. https://nl.mathworks.com/help/simulink/slref/anti-windup-control-using-a-pid-controller.html. Accessed: 2020-01-24
4. Matlab simulink model 2. https://nl.mathworks.com/help/simulink/slref/simulating-automatic-climate-control-systems.html. Accessed: 2020-01-24
5. Matlab simulink model 3. https://nl.mathworks.com/help/simulink/slref/simulation-of-a-bouncing-ball.html. Accessed: 2020-01-24
6. Matlab simulink model 4. https://bit.ly/simulinkModel4. Accessed: 2020-01-24
7. Matlab simulink model 5. https://bit.ly/simulinkModel5. Accessed: 2020-01-24
8. Matlab simulink model 6. https://bit.ly/simulinkModel6. Accessed: 2020-01-24
9. Matlab simulink model 7. https://bit.ly/simulinkModel7. Accessed: 2020-01-24
10. Matlab simulink model 8. https://nl.mathworks.com/help/simulink/slref/designing-a-guidance-system-in-matlab-and-simulink.html. Accessed: 2020-01-24
11. Matlab simulink model 9. https://bit.ly/simulinkModel9. Accessed: 2020-01-24
12. Uml - class diagram 1. http://models-db.com/repository/70/classdiagram/238. Accessed: 2020-01-24
13. Uml - class diagram 2. https://www.gituml.com/viewz/5. Accessed: 2020-01-24
14. Uml - class diagram 3. https://www.gituml.com/viewz/87. Accessed: 2020-01-24
15. Uml - class diagram 4. https://www.gituml.com/viewz/26. Accessed: 2020-01-24
16. Uml - class diagram 5. https://www.gituml.com/viewz/27. Accessed: 2020-01-24
17. Uml - class diagram 6. https://www.gituml.com/viewz/20. Accessed: 2020-01-24
18. Uml - class diagram 7. http://models-db.com/repository/84/classdiagram/441. Accessed: 2020-01-24
19. Uml - class diagram 8. http://models-db.com/repository/84/classdiagram/449. Accessed: 2020-01-24
20. Uml - class diagram 9. http://models-db.com/repository/102/classdiagram/624. Accessed: 2020-01-24

21. Uml - sequence diagram 1. http://models-db.com/repository/108/classdiagram/781. Accessed: 2020-01-24
22. Uml - sequence diagram 2. http://models-db.com/repository/108/classdiagram/783. Accessed: 2020-01-24
23. Uml - sequence diagram 3. http://models-db.com/repository/108/classdiagram/808. Accessed: 2020-01-24
24. Uml - sequence diagram 4. http://models-db.com/repository/108/classdiagram/809. Accessed: 2020-01-24
25. Uml - use case diagram 1. http://models-db.com/repository/108/classdiagram/733. Accessed: 2020-01-24
26. Uml - use case diagram 2. http://models-db.com/repository/108/classdiagram/734. Accessed: 2020-01-24
27. Uml - use case diagram 3. http://models-db.com/repository/108/classdiagram/736. Accessed: 2020-01-24
28. Uml - use case diagram 4. http://models-db.com/repository/108/classdiagram/775. Accessed: 2020-01-24
29. The uml repository. http://models-db.com. Accessed: 2020-01-23
30. Ai, B., Sentis, L., Paine, N., Han, S., Mok, A., Fok, C.L.: Stability and performance analysis of time-delayed actuator control systems. Journal of Dynamic Systems, Measurement, and Control **138**(5) (2016)
31. Atkinson, C.: Orthographic software modelling: a novel approach to view-based software engineering. In: European Conference on Modelling Foundations and Applications, pp. 1–1. Springer (2010)
32. Benjamini, Y., Hochberg, Y.: Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society. Series B (Methodological) **57**(1), 289–300 (1995). https://doi.org/10.2307/2346101. URL http://dx.doi.org/10.2307/2346101
33. Hebig, R., Giese, H., Stallmann, F., Seibel, A.: On the complex nature of mde evolution. In: International Conference on Model Driven Engineering Languages and Systems, pp. 436–453. Springer (2013)
34. Herbert, H.: The history of ocr, optical character recognition. Manchester Center, VT: Recognition Technologies Users Association (1982)
35. Ho-Quang, T., Chaudron, M.R., Samúelsson, I., Hjaltason, J., Karasneh, B., Osman, H.: Automatic classification of uml class diagrams from images. In: 2014 21st Asia-Pacific Software Engineering Conference, vol. 1, pp. 399–406. IEEE (2014)
36. Kaliappan, V.K., Yong, H., Dugki, M., Choi, E., Budiyono, A.: Reconfigurable intelligent control architecture of a small-scale unmanned helicopter. Journal of Aerospace Engineering **27**(4), 04014,001 (2014)
37. Karasneh, B., Chaudron, M.R.: Extracting uml models from images. In: 2013 5th International Conference on Computer Science and Information Technology, pp. 169–178. IEEE (2013)
38. Karasneh, B., Chaudron, M.R.: Img2uml: A system for extracting uml models from images. In: 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, pp. 134–137. IEEE (2013)
39. Melo, C.A.B., Dueire Lins, R.: A comparative study on ocr tools. In: Vision Interface (1999)
40. Modi, H., Parikh, M.: A review on optical character recognition techniques. International Journal of Computer Applications **160**(6), 20–24 (2017)
41. Moreno, V., Génova, G., Alejandres, M., Fraga, A.: Automatic classification of web images as uml diagrams. In: Proceedings of the 4th Spanish Conference on Information Retrieval, pp. 1–8 (2016)

42. Mustafiz, S., Denil, J., Lúcio, L., Vangheluwe, H.: The ftg+ pm framework for multi-paradigm modelling: An automotive case study. In: Proceedings of the 6th International Workshop on Multi-Paradigm Modeling, pp. 13–18 (2012)
43. Rashid, S.: Automatic classification of uml sequence diagrams from images (2019)
44. Reis, A., Paulino, D., Filipe, V., Barroso, J.: Using online artificial vision services to assist the blind - an assessment of microsoft cognitive services and google cloud vision. In: Á. Rocha, H. Adeli, L.P. Reis, S. Costanzo (eds.) Trends and Advances in Information Systems and Technologies, pp. 174–184. Springer International Publishing, Cham (2018)
45. Stahl, T., Voelter, M., Czarnecki, K.: Model-driven software development: technology, engineering, management. John Wiley & Sons, Inc. (2006)
46. Sun, Y., Gray, J., Bulheller, K., von Baillou, N.: A model-driven approach to support engineering changes in industrial robotics software. In: International Conference on Model Driven Engineering Languages and Systems, pp. 368–382. Springer (2012)
47. Tovar-Arriaga, S., Vargas, J.E., Ramos, J.M., Aceves, M.A., Gorrostieta, E., Kalender, W.A.: A fully sensorized cooperative robotic system for surgical interventions. Sensors **12**(7), 9423–9447 (2012)
48. Vijayarani, S., Sakila, A.: Performance comparison of OCR tools. International Journal of UbiComp **6**(3), 19–30 (2015)
49. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)