Object Detection in Design Diagrams with Machine Learning*

Jukka K. Nurminen[1], Kari Rainio[1], Jukka-Pekka Numminen[2], Timo Syrjänen[2], Niklas Paganus[3], and Karri Honkoila[3]

[1] VTT Technical Research Centre of Finland, Espoo, Finland
{jukka.k.nurminen,kari.rainio}@vtt.fi
[2] Pöyry Oy, Vantaa, Finland {jukka-pekka.numminen,timo.syrjanen}@poyry.com
[3] Fortum Oy, Espoo, Finland {niklas.paganus,karri.honkoila}@fortum.com

Abstract. Over the years companies have accumulated large amounts of legacy data. With modern data mining and machine learning techniques the data is increasingly valuable. Therefore being able to convert legacy data into a computer understandable form is important. In this work, we investigate how to convert schematic diagrams, such as process and instrumentation diagrams (P&I diagrams). We use modern machine learning based approaches, in particular, the Yolo neural network system, to detect high-level objects, e.g. pumps or valves, in diagrams which are scanned from paper archives or stored in pixel or vector form. Together with connection detection and OCR this is an essential step for the reuse of old planning data.
Our results show that Yolo, as an instance of modern machine learning based object detection systems, works well with schematic diagrams. In our concept, we use a simulator to automatically generate labeled training material to the system. We then retrain a previously trained network to detect the components of our interest. Detection of large components is accurate but small components with sizes below 15% of page size are missed. However, this can be worked around by dividing a big diagram into a set of smaller subdiagrams with different scales, processing them separately, and combining the results.

Keywords: Object detection · Schematic diagrams · Machine Learning · Legacy data.

1 Introduction

Over the years companies have accumulated large amounts of legacy data. The data can contain useful information but is difficult to take advantage of because the data is typically in paper or picture form. The goal of our research is to study ways to detect high-level objects from drawing data. As an example, a process and instrumentation diagram (P&I diagram) in drawing form is just a set of points in pixel representation, or a set of graphical objects, like lines, arcs,

* This work was partly funded by Business Finland Engineering Rulez project.

or polylines, in vector presentation. To be useful this level is not enough. Instead we need to detect if the diagram contains higher level elements, such as pumps and valves (see Fig 1). Also the connections between the higher level elements are important as well as the related texts. However, in this work we focus only on the detection of higher level elements. Text detection and recognition (OCR) as well as connection detection are handled with different algorithms.

Detecting high-level objects is useful for multiple reasons. First, there is a lot of excitement about AI and machine learning. A key ingredient for machine learning is data. To be useful for training machine learning systems, the data has to be at a higher conceptual level. Second, legacy data is often useful for human experts, but finding the relevant data, e.g. drawings of a specific aspect of legacy plants, can be difficult. Therefore, systems which allow users to make high-level queries, e.g. searching for drawings which have specific component configurations, are useful. Third, maintenance and upgrade of old plants (brownfield projects) can be executed more efficiently with fewer errors and lower risk if old plans in digitized form can be used as a basis. In summary, to unleash the value of legacy data, it is important to make its use easier and high-level object detection is one key enabler towards this goal. In the present work, our focus is P&I diagrams of process industry but similar issues are encountered in all kinds of two-dimensional diagrams, such as electrical circuits.

The problem of detecting high-level objects in design diagrams is not easy. The interesting symbols can appear in different sizes. They can also be rotated. Often it is not apparent which lines in the diagram are part of the object symbol and which lines are connectors or related to accompanying text. Therefore it is far easier to detect lines and other graphical primitives than the higher level objects. Old paper drawings in archives may also have corrections, stains, and other aspects which complicate the detection. In large diagrams individual components can also be very small in size.

The problem is well understood and there are various attempts to solve it. The simplest approach is to rely on human effort to convert old diagrams into an intelligent form. Some researchers suggest different kind of algorithms for the task (more details in Section 2). Others develop ad hoc systems combining multiple ideas for practical solutions, e.g. [20, 7, 3]. In general, it seems that, in spite of over 30 years of effort, there is still no good general solution to the problem.

A disadvantage of many of the approaches is that they need to be tailored to the task at hand, which makes their adoption expensive. The attractiveness of a machine learning based solution is that, ideally, the adaptation of the approach is easy. Just retrain the machine learning system to recognize new kinds of symbols. Another opportunity for machine learning is that the system can incrementally improve its detection capability by the modifications an expert makes manually to correct the machine suggested recognized components.

Machine learning has made rapid progress during the last years. One of the key new insights in supervised learning has been to avoid an extensive feature engineering phase. Earlier, experts, using domain-specific knowledge and logical

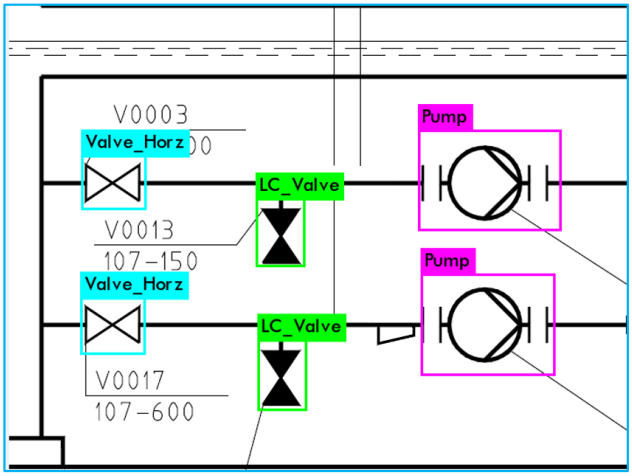


Fig. 1. Example of object detection in a P&I diagram

thinking, tried to come up with the key features to make the detection successful. Instead, modern machine learning relies on huge neural networks with millions of parameters to let the system itself learn the key features. This has shifted the emphasis of machine learning to massive datasets and their use for training. Another enabler is a possibility to retrain a previously trained network. It is possible to take a network that has been trained with massive amounts of data to detect the key patterns relevant for the detection problem and use a quite small amount of data to retrain it to detect the symbols interesting for a certain use case [16].

In our work, we use Yolo, which is a state-of-the-art neural network based system for real-time object detection [17]. The design goal of Yolo has been realtime detection of objects in video streams. It can analyze tens of pictures in a second making the algorithm useful e.g. for autonomous vehicle solutions where the speed is essential. For our needs, Yolo is an overkill. Time is not hugely critical for analyzing old diagrams. Our 2D drawing data is much simpler than 3D video feeds. On the other hand, Yolo is readily available and if it works well with our object detection tasks it can be used as such. If we easily can retrain it to detect all kinds of diagram symbols it has potential to be a versatile, adaptable tool for object detection in legacy diagrams.

In comparison to object detection in a picture of regular 3D scenes the detection of schematic symbols is different

- Objects do not form regions of similar shades of color. Instead, they are just areas defined by black lines on white background. (Sometimes also other colors are used).
- Detection time does not matter unlike in video feed analysis.

- The components are always depicted from the same perspective and the lighting conditions are stable. However, there can be extra lines and text overlapping the component symbols as well as stains and dirt.
- Lines defining components, connections, and related texts are typically of the same color. Therefore, component detection has to be made by considering the whole shape of an object rather than individual lines or regions.

In this work, we wanted to experiment and see what is possible with Yolo. In this paper we present our finding:

1. We propose the use the general purpose Yolo algorithm to high-level object detection in design diagrams.
2. We report the results of our experiments of training and using Yolo to detect objects in legacy diagrams.
3. We summarize our key learnings and observations both of training and using Yolo as well as suggest ideas for further work.

2 Related Works

The problem of converting paper-based engineering drawing to intelligent form is old. The earliest references to the problem are from the 1980s. For instance, Karima et al. [12] review the technologies available for this task. Kasturi et al. [13] present a system for detecting the symbols, their relationships, and related texts in engineering drawings. A large number of papers aims for solving the problem. Jin et al. [10] apply a mixture of techniques. Adam et al. [2] apply Fourier-Mellin transformation. A lot of the work before the year 2000 is summarized in the review by Cordella and Vento [5].

Since the turn of the century new ideas include symbol detection directly from raster images without vectorization [11], using region adjacency graphs with integer optimization [4], application of dominant chain code [6]. Ruo-yu et al. [19] propose a learning-based method where the system learns to improve its recognition capability by the modification a human user makes to fix the recognized components. Ablameyko and Uchida [1] is a review of the work in early 2000 on this area. Practice-oriented solutions, e.g. [7, 3], combine multiple approaches.

The classical approaches to detect objects is based on defining their characteristic features and them trying to match the picture object to those. Algorithms for this approach are summarized in [18].

The modern approach to object detection is based on supervised machine learning. Especially the object detection in video streams has attracted a lot of research. It has many use cases of significant importance, especially in autonomous driving. Therefore there has been rapid development in this area lately.

In supervised machine learning, the system is given a set of examples with known object names and their bounding boxes. The core of the system is a neural network with a vast number of parameters. The system learns the values of these parameters by trying to minimize the error in its estimations. Different machine

learning systems vary in the way the neural network is structured and how the problems are represented. RCNN [22] and its variants first find regions in the picture and only then try to detect the object in each region. For schematic diagrams this is a problematic approach and finding regions in a black and white line drawing is problematic. The Yolo algorithm we use finds and classifies regions in single pass [17]. A recent detailed discussion of different machine learning based object detection algorithms is available in [14]. For our purposes the details of the algorithm do not matter much as long as the overall concept of training it with example data is not changed. Replacing Yolo with more efficient future algorithm should be simple.

3 Yolo for Object Detection in Engineering Drawings

To train Yolo to detect objects we started with a pre-trained Yolo model. The essential idea is that with pre-training it already knows how to detect essential features for object detection. With a small amount of training data we can then retrain it to detect just those components we are interested in. As Yolo has over 10 million parameters it would require a huge amount of computing and data to train it from scratch.

Having enough labeled training material, i.e. pictures with objects and their known types and bounding boxes, is frequently a problem. In our case, we came up with a way to generate training material automatically. We used Apros simulator [4] to generate the training data. Apros is intended for dynamic simulation of power plants, energy systems, and industrial processes. However, it supports user-defined components with custom symbols. We provided it with an example of each interesting component and with a small script it was then able to generate training data which contained the components in different sizes, rotations, and locations, their names and bounding boxes. This was a convenient way to rapidly generate labeled training material, a task which is commonly considered difficult in supervised machine learning.

We performed two kinds of experiments. First, we investigated Yolo training with artificial examples generated by Apros. Then, we used the component library and diagrams of Pöyry engineering company to see how the system works with real legacy data. In both experiments we used the same environment consisting of: (i) yolov3-tiny with pre-trained weights, (ii) Azure virtual machine (Standard F8 virtual machine; 8 vCPUs, 16 GB memory; Later NC6 with GPU support; 6 cores, 56 GB memory) with Ubuntu version Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-1032-azure x86_64), (iii) Apros simulator.

3.1 Artificial Apros Experiments

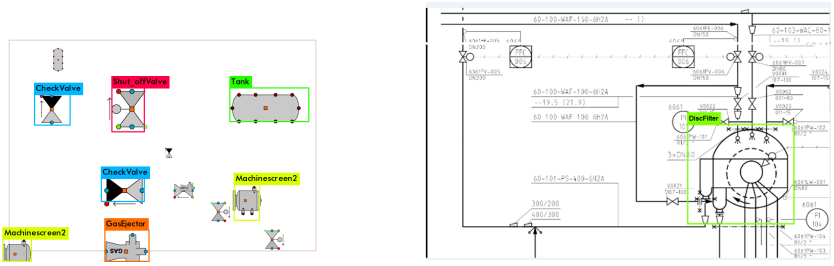
For these experiments we generated 2000 random Apros diagrams. To save training time we used only seven different component types. We selected these com-

[4] http://www.apros.fi/en/

ponents with the aim to have some which are common and have simple symbols (e.g. ControlValve and CheckValve), some with more complicated symbols (e.g. Machinescreen2, LiquidEjector, GasEjector), and some with bigger symbols (Tank). The idea was to try out how the symbol size, complexity, and similarity to other symbols influence the detection.

Yolo recommends having at least 300 pictures of each object for training. To ensure this we generated 2000 random diagrams each with max 12 components. The optimum number of training examples was not in the scope of our present work but in the future it would be useful to study if a smaller amount of training examples would be adequate as the difference in symbols in 2D schematic drawings is far smaller than in video streams of 3D objects.

We performed multiple training experiments with (i) rotated and mirrored components, (ii) different component sizes, and (iii) partly visible components (components overlapping each other or located partly outside of picture area).



(a) Artificial test diagram (b) Real P&I diagram

Fig. 2. Examples of detecting symbols from design diagrams

An example result is visible in Fig 2(a). The general observation is that the system is able to detect large component well. It is also able to deal with overlapping and only partly visible components as well as with rotated and mirrored components.

However, the size of the components has significant influence to the detection. It seems that components smaller than around 15% of the screen width or height are poorly detected. As can be seen in Fig 2 the system is good at detecting the bigger components but fails to detect the smaller ones. The reason for this is that by default Yolo uses 416x416x3 grid. When it scales small components to that size they lose a lot of their characteristic features and become so blurry that classifying them correctly is difficult even for a human.

To deal with this problem we have two main options. First, change the default width and height of Yolo resolution to e.g. 832x832. This increases of the neural network model a lot requiring more training and it still may not be good enough.

Alternatively, we can split the bigger diagram into smaller rectangular regions and zoom them out to make the components big enough for Yolo to de-

tect. This assumes the resolution of the original figures is good enough to allow this kind of scaling. If the data is in vector graphics format this should be fine. For scanned data scaling is not so easy but with a high resolution scanner the diagrams should tolerate some zooming. In this approach Yolo needs to make multiple rounds to detect the components in each rectangular subdiagram. Because the detection speed of Yolo is very fast running it multiple times is not a problem. Some additional software is needed to manage the process of splitting, detecting, and combining the detection results. Naturally, this adds complexity to the component detection because the system needs to decide how to split the diagrams, which zooming ratios to apply, and how much overlap the different regions should have. Further research on making object detection algorithms better in dealing with small objects is therefore welcome as well as solutions, which automatically detect good ways to split and scale diagrams.

3.2 Experiments with Real Pöyry Data

To validate the concept with real-world data we used components from Pöyry component library. Altogether it contains around 800 different symbols. However, we limited ourselves to 200 most commonly used symbols.

We converted these symbols to Apros custom components and generated 20.000 random diagrams for training. We then tested the trained network with legacy diagrams from past Pöyry projects. Because of the large number of components and needed training instances we used Azure instances which provided support for GPUs. The use of GPUs has a dramatic effect on training speed with approximate 50x speed increase for training. We ran 250.000 training iterations which took about 24 hours.

Fig 2(b) shows an example how the system can detect the relevant component in a complicated real design diagram. It can find the essential features and is not mislead by the incoming connections or overlapping text.

At the same time, the figure shows that in this scale the system is not able to detect the smaller components (e.g. valves above the DiscFilter) because they are too small. Therefore zooming-in to the picture and finding the components with different resolutions is essential.

4 Learnings

Small components are problematic but we can deal with them by dividing the diagram into smaller rectangles which have bigger components. This requires that the resolution of the original diagram tolerates zooming to make the components big enough in the separate diagrams. A rough rule of thumb is that each component should be around 15% of the height or width of the subdiagram.

Detecting the components works well once they are big enough. Also, the system is able to detect their bounding boxes very accurately.

Detecting small objects is a common weakness in object detection systems [15]. Some of the solutions are based on dividing the original figure into smaller

subfigures and using them at different scales [15]. Another approach would be to just use Yolo to detect only large components. In the P&I diagrams smaller components, e.g. pumps and valves, have simple fixed-size symbols. Therefore they can be detected with straightforward template matching. In our case we tried this to find pumps and valves with OpenCV template matching `matchTemplate`. Because `matchTemplate` is only able to detect components of the same size we first used Tesseract OCR engine [21] to detect text and then used the text size to scale the template for `matchTemplate`. This approach works as long as the diagrams have been created with systematic rules for text and object sizes. For more flexible detection, scaling is needed. Future innovations in small object detection may provide better algorithms as well.

Depending on HW the training times vary but it will require around 20.000 iterations to show good results. Looking at the progress of learning is interesting. In early phase at smaller iteration amounts Yolo's estimates for the confidence of correct detection are small. As the training progresses the confidence grows and for many components is close to 100% in the end. Likewise, the bounding box sizes become increasingly accurate during the training. With a large number of iterations the progress stabilizes although even after 100.000 iterations some small progress was still done.

The amount of manual effort and expertise to train the system to detect new symbols is small. All that is needed are one example picture of each component together with its bounding box data. The rest is handled automatically: Apros generates the training material and Yolo uses it to train the network parameters to appropriate values. In our test some manual steps were needed to transfer data and invoke the different tools but also those steps can be automated with appropriate scripts.

In general, Yolo seems to be doing well in this task, even if is completely outside of its target domain. There is heavy competition to find improved image detection algorithms in machine learning (see e.g. [8, 9] for comparison of other recent algorithms). Therefore, it is likely that we will soon see even better performing alternatives.

5 Conclusions

In this paper we have investigated the problem of converting schematic drawings into an "intelligent form". With intelligent we here mean that the system is able to understand what objects are in the drawing and how they are connected. This is an essential step for making the legacy data useful for data mining, machine learning, and brownfield design tasks.

We have used a pre-trained Yolo neural network and used data generated with our own simulator to retrain the network to detect the components we are interested in. The experiments were done with process and instrumentation diagrams of process industry but the same approach works equally well for other kinds of schematic diagrams such as drawings of electrical circuits.

The key finding is that the Yolo approach, which has been intended originally for entirely different purpose, detection of objects in video streams, works well also in schematic diagrams. It has problems in detecting small components, but this problem can be overcome by letting it detect zoomed-in parts of the picture. Because the detection speed of Yolo is very fast (multiple frames per second) we can easily afford to run tens or even hundreds of separate object detection runs to subparts of the original diagram with different zooming ratios.

Besides object detection we have also evaluated the other subtasks needed for legacy data use (detection of connections and detection and OCR of text). In our future work we plan to put the different pieces together to create a system which can process legacy diagrams and convert them automatically into a standard form such as DEXPI. Then they can be used for different kind of tasks allowing companies to gain value from their legacy data.

References

1. Ablameyko, S.V., Uchida, S.: Recognition of Engineering Drawing Entities: Review of Approaches. International Journal of Image and Graphics **07**(04), 709–733 (10 2007). https://doi.org/10.1142/S0219467807002878, http://www.worldscientific.com/doi/abs/10.1142/S0219467807002878
2. Adam, S., Ogier, J., Cariou, C., Mullot, R., Labiche, J., Gardes, J.: Symbol and character recognition: application to engineering drawings. International Journal on Document Analysis and Recognition **3**(2), 89–101 (12 2000). https://doi.org/10.1007/s100320000033, http://link.springer.com/10.1007/s100320000033
3. Arroyo, E., Hoernicke, M., Rodríguez, P., Fay, A.: Automatic derivation of qualitative plant simulation models from legacy piping and instrumentation diagrams. Computers & Chemical Engineering **92**, 112–132 (2016)
4. Bodic, P.L., Locteau, H., Adam, S., Héroux, P., Lecourtier, Y., Knippel, A.: Symbol Detection Using Region Adjacency Graphs and Integer Linear Programming. In: 2009 10th International Conference on Document Analysis and Recognition. pp. 1320–1324. IEEE (2009). https://doi.org/10.1109/ICDAR.2009.202, http://ieeexplore.ieee.org/document/5277721/
5. Cordella, L., Vento, M.: Symbol recognition in documents: a collection of techniques? International Journal on Document Analysis and Recognition **3**(2), 73–88 (12 2000). https://doi.org/10.1007/s100320000036, http://link.springer.com/10.1007/s100320000036
6. De, P., Mandal, S., Das, A., Bhowmick, P.: A New Approach to Detect and Classify Graphic Primitives in Engineering Drawings. In: 2014 Fourth International Conference of Emerging Applications of Information Technology. pp. 243–248. IEEE (12 2014). https://doi.org/10.1109/EAIT.2014.33, http://ieeexplore.ieee.org/document/7052053/
7. Henderson, T.C.: Analysis of engineering drawings and raster map images. Springer (2014)
8. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: IEEE CVPR. vol. 4 (2017)

9. Hui, J.: Object detection: speed and accuracy comparison (faster r-cnn, r-fcn, ssd, fpn, retinanet and yolov3). https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359/ (2018)
10. Jin, L., Zhou, Z., Xiong, S., Chen, Y., Liu, M.: Practical Technique in Conversion of Engineering Drawings to CAD Form. IMTC/98 Conference Proceedings. IEEE Instrumentation and Measurement Technology Conference. Where Instrumentation is Going (Cat. No.98CH36222) **1**, 8–13 (1998). https://doi.org/10.1109/IMTC.1998.679631, http://ieeexplore.ieee.org/document/679631/
11. Jiqiang Song, Feng Su, Chiew-Lan Tai, Shijie Cai: An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(8), 1048–1060 (8 2002). https://doi.org/10.1109/TPAMI.2002.1023802, http://ieeexplore.ieee.org/document/1023802/
12. Karima, M., Sadhal, K., McNeil, T.: From Paper Drawings to Computer-Aided Design. IEEE Computer Graphics and Applications **5**(2), 27–39 (1985). https://doi.org/10.1109/MCG.1985.276400, http://ieeexplore.ieee.org/document/4056067/
13. Kasturi, R., Bow, S., El-Masri, W., Shah, J., Gattiker, J., Mokate, U.: A system for interpretation of line drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(10), 978–992 (1990). https://doi.org/10.1109/34.58870, http://ieeexplore.ieee.org/document/58870/
14. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. arXiv preprint arXiv:1809.02165 (2018)
15. Meng, Z., Fan, X., Chen, X., Chen, M., Tong, Y.: Detecting small signs from large images. In: Information Reuse and Integration (IRI), 2017 IEEE International Conference on. pp. 217–224. IEEE (2017)
16. Pan, S.J., Yang, Q., et al.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2010)
17. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv (2018)
18. Roth, P.M., Winter, M.: Survey of appearance-based methods for object recognition. Inst. for Computer Graphics and Vision, Graz University of Technology, Austria, Technical Report ICGTR0108 (ICG-TR-01/08) (2008)
19. Ruo-yu, Y., Feng, S., Tong, L.: Research of the structural-learning-based symbol recognition mechanism for engineering drawings. Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference on pp. 346–349 (2010)
20. Smith, R.: An Overview of the Tesseract OCR Engine. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2. pp. 629–633. IEEE (9 2007). https://doi.org/10.1109/ICDAR.2007.4376991, http://ieeexplore.ieee.org/document/4376991/
21. Smith, R.: An overview of the tesseract ocr engine. In: Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. vol. 2, pp. 629–633. IEEE (2007)
22. Uijlings, J.R., Van De Sande, K.E., Gevers, T., Smeulders, A.W.: Selective search for object recognition. International journal of computer vision **104**(2), 154–171 (2013)