Object detection in construction department

Pralipa Nayak
Machine learning Developer
pralipan@vconstruct.in

Abstract:

The capability to automatically identify shapes and objects from the image content through direct and indirect methodologies has enabled the development of several civil engineering related applications that assist in the development of construction projects. This paper shows how this automation process has been made possible using Machine Learning techniques to reduce the time and effort consumed in the field of construction industry. Also how technology has helped to transform the construction industry. Manually where various objects like doors, beams and columns were detected throughout an architectural plan by counting and recording, this process used to take many hours to work on a single plan. Automating this process with the help of Object detection and Machine learning has helped to reduce the time and effort consumption by a huge margin.

Introduction:

In the construction industry, most of the work, from verifying an architectural plan to detecting various objects in the plan, are done manually. This in turn takes a lot of time and effort.

Object detection is a computer vision technique that can be used to automate this process which in turn, will reduce time and effort to a great extent. It basically has a series of instructions for a computer to transform input data into desired output. Instructions are mostly based on an IF-THEN structure: when conditions are met, the program executes a specific action.

Machine learning techniques like Image processing can be used to further enhance this process and perform object detection with better accuracy and speed. It enables machines to solve problems with little or no human input., and take actions based on past observations. This approach will help to automate detecting various objects in an architectural plan and various other work and help us save time and effort.

Background:

We are surrounded by different types of beautiful architecture around us. And the basis for any architecture is the architectural plan. For any architecture to stand firmly for ages, the plan should be perfect with negligible to no flaws.

Object detection helps to a great extent in recognizing designs according to existing data and extracting data from them for further investigations like finding flaws or whether any section needs changes, detecting changes in revised plans in accordance with old plans and quantifying elements in a plan.

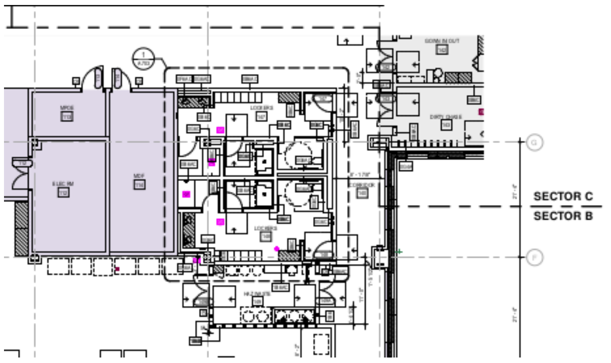

Figure 1. Image of an architectural plan from which doors and water closets are to be recognized.

The above given figure is an architectural plan, based on which construction will be made. Our goal was to detect various objects like doors, water closets, columns, beams etc.

Approach:

On-Screen Takeoff is a construction estimating and takeoff solution for contractors and construction professionals. It is widely used to mark items on a 2d plan and add properties to the items. Once the relevant items are marked, it can extract the desired quantities as required. But this is a manual process and is repetitive in all projects. On an average, it requires nearly a day for a single plan. Another tool called Autodesk Takeoff also does the same thing, however it also takes a similar amount of time.

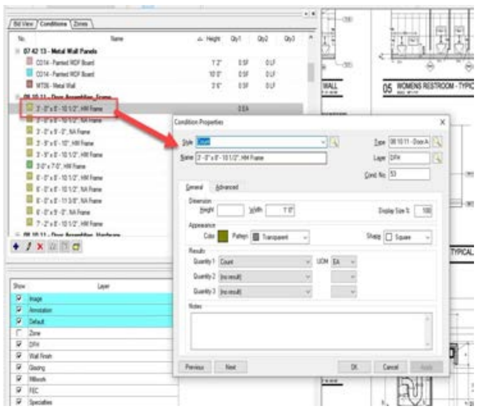

Figure 2. Image showing details of a page in On-screen Takeoff.

Figure 2 shows the details of the architectural pan in an indexed way. Using OST(On-screen Takeoff) formulas are prepared to mark different objects like doors, water closets, etc one by one. The objects are then marked throughout all the pages as per the formulas manually. This method takes up lots of time and effort.

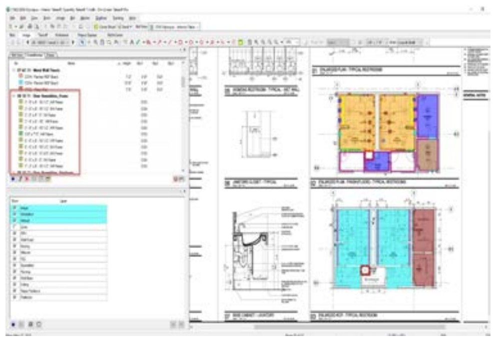

Figure 3. Image showing marked doors in the architectural plan.

As per figure 3, we can see the doors have been marked as per the formulas created however it was done one by one for all the objects. All the similar objects will be marked the same way.

The next approach was to use machine learning techniques to make the process automated. This way we were able to get the data in a very less amount of time as compared to the manual approach. Machine learning helped us to get the data within an hour which saved us lots of time. We had a pdf which has multiple pages of architectural plans. Machine learning techniques like pdf mining and image processing were used to achieve the results.
Pdf mining helped in reading and extracting details from every page.

Challenges:

The first approach for ML was using OpenCV. The challenges faced were:

1. This approach detects images exactly the same as template images.
2. This approach does not support rotation and resizing of objects so to solve the rotation issue the template image was rotated 90 degree clockwise, 180 degree and 90 degree counterclockwise then all the rotated template images were matched to the input image and the output was found, however the size of the object in input image has to be exact with the template image size and if one object in input image is rotated by 95 degree, then OpenCV won't consider this as a match.
3. Accuracy was extremely low and it did not support overlapping.

The next approach was using Tensorflow. The challenges faced were:

1. The accuracy was very good (above 95%), however it did not work with larger images(2-4 MB) which did not satisfy the requirement.

The next approach was Custom-Vision. The challenges faced were similar to the former approach.

1. Though it supports overlapping and accuracy was also very good(above 95%), it did not work with large images(2-4MB) which did not satisfy the requirement.

Solution:

Then we moved towards using YoloV5 (You Only Look Once).

As per machine learning standards, first train and test datasets were prepared by splitting in 80-20 ratio and labelled using LabelImg library.

After labelling, the model was trained using YoloV5 (You Only Look Once) and we got the trained model for detecting doors.

Then we converted the pdf page to an image file in which object(door) detection was performed.

As a 2-d architectural plan is way too large, we used a library called SAHI to detect objects in large images. SAHI stands for Small object detection by slicing Aided Hyper Interface. It is a vision library for large scale object detection and instance segmentation. What SAHI basically does is, it slices large images into smaller segments and performs object detection on those segments so it easily picks almost all the objects in a large image.

We created a script written in python3 that loads the trained model, takes up the pdf file as an input parameter, reads each page as an image, then uses SAHI to slice the image and perform object detection on the image with pretty good accuracy.

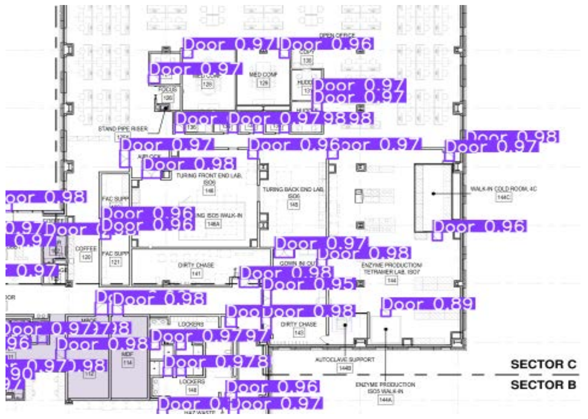

Figure 4. Image with detected doors and labeled accordingly.

The above image is one of the results we got after performing object detection on a given pdf file. The detected doors haven been labelled as doors at their respective coordinates. This gives us a clear idea of which of the elements present in the image (a page in pdf) are doors or any other elements.

Similar process was followed for detecting other objects. Model was trained accordingly using related datasets and prediction was made based on the trained model.

Another scenario popped where we needed written textual data from pdf files consisting of tables. This seemed easy as there are lots of packages like Camelot, PyPdf that deal with such scenarios. However, they are limited to text pdfs. The need was to extract those data from image/ scanned pdfs. Image processing helped here too. Pytesseract or Python-tesseract was used to get the results. Pytesseract is an Optical Character Recognition tool for python. It reads and recognizes text in images.

The challenge now was to provide the coordinates or the location from where pytesseract will perform the recognition. Since in a table, the data are separated to groups using vertical and horizontal lines, those lines were extracted first. It was done by inverting the image first to monochrome. Then the vertical line and horizontal lines were extracted from the table, which helped us get the coordinates from the image. The co-ordinates were then used to verify the locations from where we needed the data and were given to pytesseract, which read and recognized the text from those locations and gave us the data. This flow was achieved by bundling pytesseract with some data structure logic, used for manipulating the data in required structure.

The above mentioned flow is also used to extract data from the earlier mentioned 2d architectural plans like beams and columns data. Since they are also made as horizontal and vertical lines, this algorithm helps extract data like the length, angles, and thickness of the parameters. Which is required for major calculations while moving with a construction project.

Pdf files with data stored in the form of tables but where pages are images instead of plain texts are used. Here the image was processed with pytesseract to extract the data from it. First it was inverted to monochrome format and horizontal lines were extracted from the inverted image. Then from the inverted image, vertical lines were extracted. Both extracted images were then merged which gave us an inverted image of the whole table.

After the horizontal and vertical lines were merged to get the whole table in inverted format, the resulting image was then used to contour the image, Figure 5.

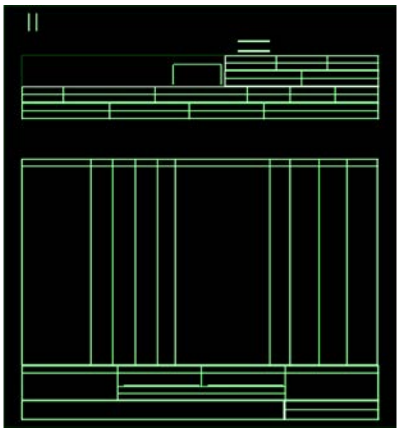

Figure 5. Table after contouring.

The coordinates were then used to recognise the smaller rectangles. And then from those rectangles, the data from the table were extracted based on some keywords like the column names. Thus data was successfully extracted from the image pdf with zero errors.

Conclusion:

The algorithms created were tested rigorously on various types of data. Which in turn gave us lots of error and opportunity at the same time to resolve them and get the algorithms ready for use with negligible to zero failures or error results.

The manual approach, using OST, gave results while taking a longer period of time, almost more than 15 hours, with involvement of human resources. This problem was solved using Machine learning where the process took just some minutes also without requiring any human efforts. This paper shows how this automated process can save us a huge amount of time and effort as compared to manual approaches. If implemented on a larger scale, this process can save a lot of time and effort, which in turn can save a lot of funds.

References:

[1] Fitz
https://pymupdf.readthedocs.io/en/latest/module.html
https://github.com/pymupdf/PyMuPDF/blob/master/fitz/fitz.i
[2] LabelImg
https://github.com/heartexlabs/labelImg
https://towardsdatascience.com/how-to-label-images-for-object-detection-step-by-step-7ee317f98583
https://medium.com/deepquestai/object-detection-training-preparing-your-custom-dataset-6248679f0d1d
[3] YoloV5
https://github.com/ultralytics/yolov5
https://learnopencv.com/custom-object-detection-training-using-yolov5/
https://stackabuse.com/object-detection-inference-in-python-with-yolov5-and-pytorch/
[4] SAHI
https://github.com/obss/sahi
https://medium.com/codable/sahi-a-vision-library-for-performing-sliced-inference-on-large-images-small-objects-c8b086af3b80
https://analyticsindiamag.com/small-object-detection-by-slicing-aided-hyper-inference-sahi/

Pralipa Nayak
Machine learning Developer
pralipan@vconstruct.in