Automated Wall Detection in 2D CAD Drawings to Create Digital 3D Models

C. Wei[a], M. Gupta[a] and T. Czerniawski[a]

[a]School of Sustainable Engineering and the Built Environment, Arizona State University, USA

E-mail: cwei32@asu.edu, mgupta70@asu.edu, Thomas.Czerniawski@asu.edu

Abstract –

Generating digital 3D buildings models from scratch is time consuming and labor intensive. In this paper, we present an automated detection process leveraging computer vision and the information available in 2D drawings to reduce 3D modeling time. The recognition system is limited to walls and has two parts: (1) Image classification on walls by ResNet-50 model, (2) Object Detection on walls by YOLOv3 model. The system accepts new 2D drawings and outputs parameters of recognized walls. The parameters are input into Dynamo for 3D model reconstruction. We anticipate these types of systems, which rely on 2D drawings as recognition priors, will be pivotal to the industry's transition from 2D to 3D information modalities.

Keywords –

Deep Learning; Image Classification; Object Detection; 2D Drawings; 3D Building Models

1 Introduction

Building Information Modeling (BIM) plays an important role during the entire life cycle of a building. First, Cloud hosted BIMs enhance communication by ensuring all views into the model are synchronized. Secondly, achieving better visualization allows clients to have a good understanding at each stage reducing the possibility of design changes in the future. Thirdly, performing clash detection tasks on BIM models can be cost effective and improve safety. Lastly, environmental analysis and simulation on BIM models accomplish sustainable and AI-based architectural design [1].

For projects during preconstruction phase, it is quite beneficial if 3D modeling process is efficient since it helps push construction start date forward by enhancing the collaboration among entities early on. Other projects are 3D reconstruction for as-built buildings which can assist in reconstructing historical buildings and better visualization through digital representation. In detail, doing structural analysis on 3D models of historical buildings can reach better maintenance. Moreover, taking this paper application as an example, constructing campus 3D representation can let online students have experience of looking around campus through virtual tours and even have interaction, such as reading posters on bulletin boards. To speed up the 3D modeling process with high accuracy, computer vision is a promising approach.

Computer vision is a prevailing implementation of artificial intelligence. It can achieve pattern recognition, object detection, image classification, and instance segmentation tasks on images, such as 2D drawings. This paper is experimenting with computer vision techniques in the automatic 3D model generation process to solve time-consuming manual modeling conditions. The scale and distribution of the dataset, quality of labeling and model choosing would all affect the model performance.

In our study, we proposed an object detection model to localize the vertical and horizontal walls on 2D drawings by computer vision technique. Some researchers were using semantic segmentation models to achieve this process [2, 3, 4, 8, 10], however, to make the process more efficient, we chose an object detection model which can reduce much labeling time and still achieve good performance. After we expand this work to multi-classes detection on 2D drawings, the performance metrics would include a confusion matrix and mean Average Precision (mAP).

2 Related Work

We focus on reviewing related work on how deep learning assists the process of 3D model reconstruction from 2D drawings, the benefits and applications of later usage.

2.1 Deep Learning Approach on 2D Drawings

There are lots of researchers doing recognition and segmentation of components on 2D drawings by deep learning approaches. Xiao et al. [2] cropped the original

2D drawings into smaller image dimensions for feeding into a neural network model. They manually did pixel-labeling on 300 2D drawings and implemented a transfer learning from ResNet-152. This model is pretrained on the ImageNet dataset and then executed recognition and localization on five architectural components: wall, window, door, column, stairs. Liu et al. [3] transformed rasterized images to vector-graphics representation since vector images can make further 3D reconstruction models having better visualization, being easier to manipulate and do computational analysis. They first did deep representation learning through Convolutional Neural Network (CNN) to convert a raster image to a set of junctions and pixel-wise semantics and then assembled junctions to lines and boxes by setting constraints through integer programming with a straightforward post-processing to achieve vectorization. Dodge et al. [4] utilized Fully convolutional networks (FCN) to do wall segmentation after trying different pixel strides. They conducted Faster R-CNN and Optical Character Recognition (OCR) to estimate room sizes. Mishra et al. [5] did furniture and architectural components detection on floorplans by Cascade Mask R-CNN network with CNN and Deformable Convolutional Networks (DCN) separately.

The recognition and segmentation tasks are not limited to architectural floor plans and can be more generalized to different types of buildings. Zhao et al. [6] detected structural components and grid reference by YOLO model. The structural elements include columns, horizontal beams, vertical beams and sloped beams. Kalervo et al. [7] presents a large-scale floor plan dataset, CubiCasa5K, which is carefully annotated by applying the Quality Assessment process and including 5000 samples with over 80 object classes.

In this paper, we adopt ResNet-50 and YOLOv3 in our methodology for wall recognition and segmentation tasks since both algorithms have their prevalence in pretrained models and can be easily deployed by industries' practitioners. Moreover, implementing the ResNet-50 model for classification tasks can be trained easily without increasing the training error with a large number of layers and this algorithm could help vanish gradient problems during the backpropagation process [8]. The YOLOv3 model for object detection has high speed and comprehends generalized object representation [9].

2.2 3D Model Generation and Application

Kalervo et al. [7] mentioned that the application of 3D models includes 3D real estate virtual tours and AR/VR technology. Jang et al. [8] created CityGML (City Geography Markup Language) and IndoorGML (Indoor Geography Markup Language) 3D data models by extracting indoor spatial information from floorplan images. These 3D models support a structure for 3D geospatial data integration, storage and exchange. Kippers et al. [9] claimed that 3D models can support better decision making, data analysis and scenario simulation. They integrated CityJSON and floor plan images to reconstruct the 3D model. Some researchers indicated that CityJSON is easier to use than CityGML.

Seo et al. [10] stated the application of the 3D model reconstruction process at each stage. Architectural component recognition can contribute to evacuation paths generation and evacuation distances calculation. Analysis of building energy ratings with window area ratios could be executed by automatically calculating the window and wall areas. Moreover, Using Generative Adversarial Networks (GAN) can generate much more virtual drawing images to integrate into AI-based architectural design in the future.

For indoor furniture fitting, Dodge et al. [4] performed Optical Character Recognition (OCR) to extract text information from 2D drawings which can measure each room size and compute the pixel density for fitting interior components.

3 Methodology

This section presents data generation process, training, inference pipelines and 3D model reconstruction process in Revit.

3.1 Data Creation

In this study, Arizona State University Tempe campus 2D Drawings were assembled by the University Facilities Management department. We utilized 29 sheets of size 3400 x 2200 pixels as our original dataset.

We then used the LabelMe annotation tool [11] to manually label walls' location. We drew a rectangle as a bounding box surrounding each wall so that we can get the location of each wall which includes the coordinates of the upper-left point, width and height of each bounding box.

To feed into the CNN architecture, we randomly sampled each sheet with 100 crops of 256 x 256 pixels images, and computed the labeled information to the relative position for the small crop images. In the meantime, we programmed the small cropped images to two folders depending on whether or not the image contained the wall for further training.

Ultimately, we assembled all labeled information into a csv file and got 1250 non-wall images and 1650 wall-contained images individually.

3.2 Model Training

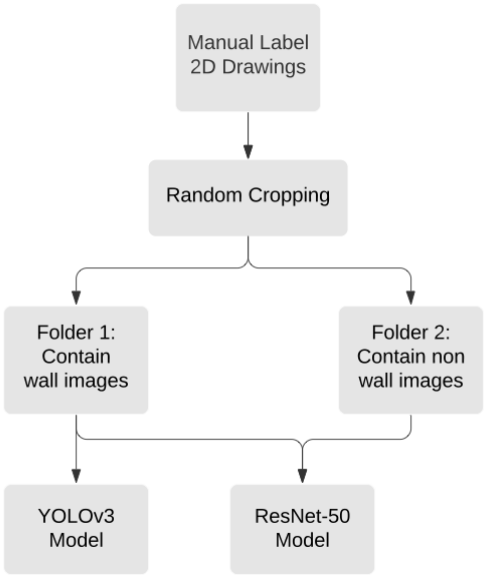

Figure 1. The training pipeline from labeling to Models' Generation

3.2.1 Wall Classification

The first step of doing model training is determining programming languages, deep learning libraries, programming environment and referred neural network models. We used Python with Tensorflow and Keras libraries on Google Colaboratory which is an online cloud-based Jupyter notebook environment. We also referred to the YOLOv3 model architecture from PyLessons [12] website which posts deep learning applications and tutorials for customized data. We chose their architecture and did further modification since the descriptions and explanations are clear, code is precise and the results they have shown are promising.

To classify wall objects, we utilized the ResNet-50 model to train a binary classification task. The training dataset was collected during the preprocessing step and a 70/30 ratio to split training and test dataset was determined. We utilized all training set as our input for the training pipeline shown in Figure 1, and test set for inference pipeline shown in Figure 3, separately.

We first programmed the ImageNet pretrained model for transfer learning. The setting of the pretrained model includes average pooling, 256 x 256 x 3 pixels input shape, 2 classes classification, and all layers inside the pretrained model are not trainable, which could reduce time complexity during the training process.

We flattened the output of the pretrained model into one dimension and added a fully connected layer with 512 nodes and a ReLU activation function. Finally, using a sigmoid function to determine the probability of wall existence.

During the training process, we set a 32 batch size, 100 epochs, and Adam optimizer with a 0.001 learning rate. The result shows that the training and test accuracy converged to 98% and 94%, respectively, as shown in Figure 2.

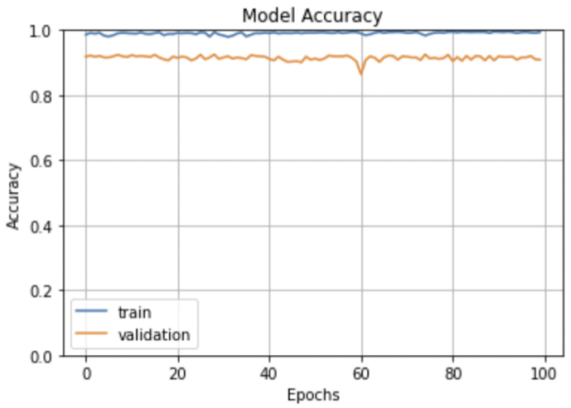

Figure 2. ResNet-50 classification training and test accuracy versus number of Epochs

3.2.2 Wall Detection

This section describes the wall detection task using the wall-contained images as our training set and YOLOv3 model for training. For this supervised learning training, we parsed the images and loaded the labeled information csv file with image filename and corresponding bounding box of wall information. We used an 80/20 ratio to split the training and test dataset, set the batch size as 4, Adam optimizer and 30 epochs in total. For learning rate, we used two warmup epochs first and then decayed linearly from 1e-4 to 1e-6.

3.3 Model Inference

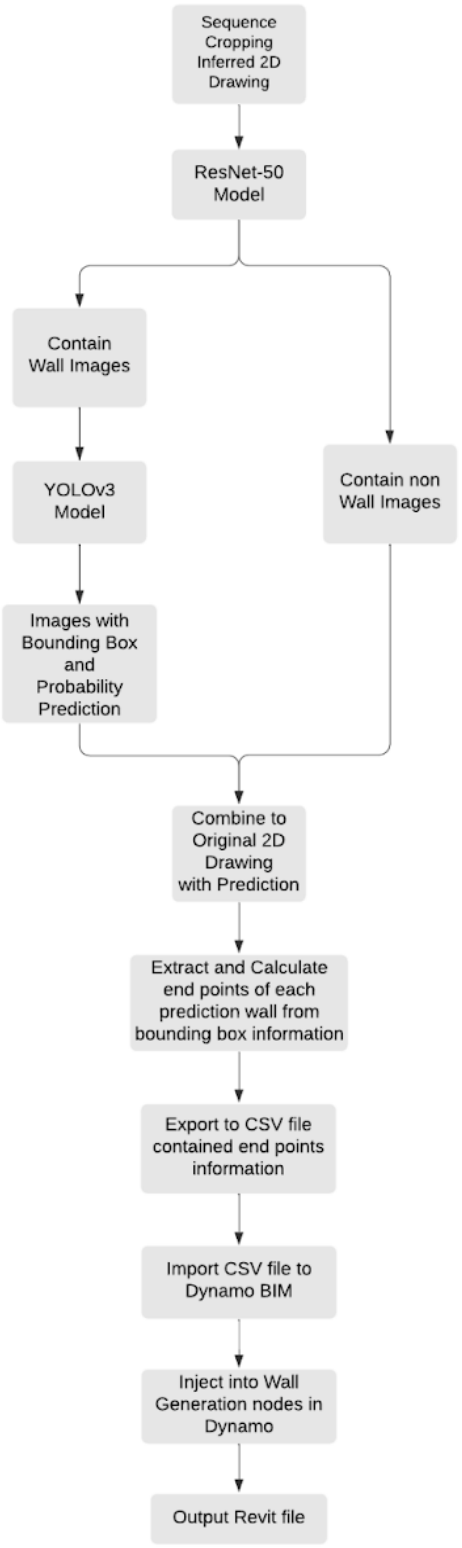

Figure 3. The inference pipeline for 3D wall reconstruction in Revit

To do model inference on new 2D drawings, we first cropped each drawing into 256 x 256 pixels size in a sliding windows sequence since we need to match the input size of deep learning algorithms that we set during the training process.

We feed each crop into our ResNet-50 model to classify if the crop contains walls or not. The inferred non-wall images stay the same since there are no walls to detect the location, however, we parsed the inferred wall images into our YOLOv3 model to infer bounding box information and probability of walls. The inferred rectangle bounding box surrounding walls and probability would be shown on each crop. Eventually, we can combine each crop to the original sheet size in a sequence after the inference process. The overall inference process is shown as Figure 3.

3.4 3D Model Reconstruction in Revit

After the model inference process described in section 3.3, we can get the prediction value of the wall location which represents the bounding boxes information. We extract and calculate the end points of each predicted wall location by their predicted bounding boxes. These parameters are saved in a csv file with height information on the elevation plan and then imported to Dynamo BIM. In the Dynamo environment, we inject the csv file into wall generation nodes and connect to the wall family type. Ultimately, we reconstruct the walls in Revit through Dynamo with our csv file and the family type we specify. The overall procedure is shown in Figure 4.

4 Results and Discussion

The results are shown in Figure 5 which are zooming a portion of the new 2D drawing. The red rectangles indicate the bounding box and the number on the upper-left corner above the rectangles is the neural network's confidence probability of the wall.

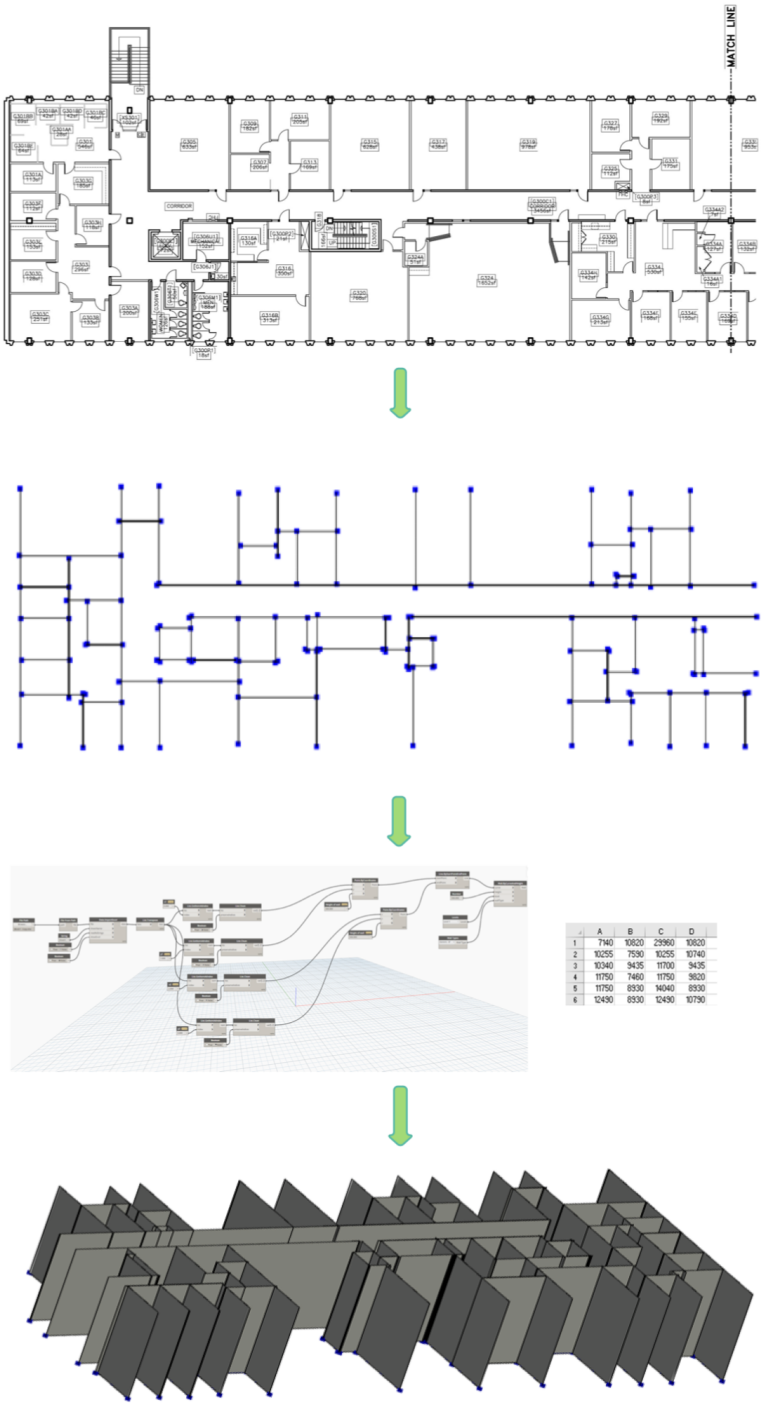

Figure 4. The procedure of creating a 3D model from extraction wall location through Dynamo and CSV file

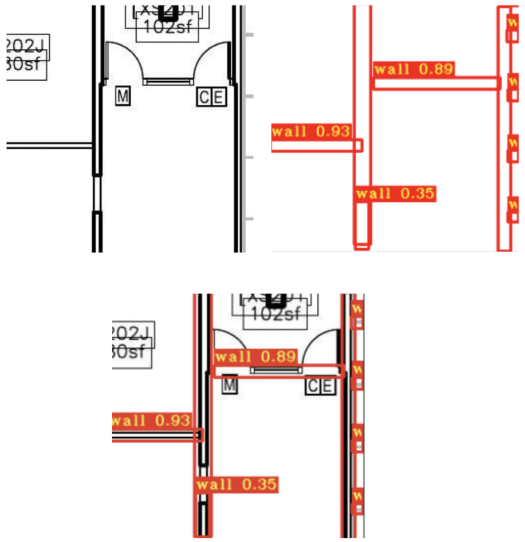

Figure 5. Original small portion of floorplan (upper-left), bounding boxes (upper-right) and floorplan with bounding boxes (lower)

We raised three failure cases on our inferred drawings as shown in Figure 6. First, the rectangle surrounding “MATCH LINE” on the drawing causes the CNN model to possibly be misinterpreted as a wall appearance. Secondly, we discovered that “curved walls” and “diagonal walls” are not suitable for object detection tasks during manual labeling process since the bounding boxes are axis aligned which could not tightly delineate the curved and diagonal lines. Therefore, we did not label the curved and diagonal walls during the training process so that our model could not recognize them on the inferred dataset.

Future work will include, first, we can remove legend, title and match line symbol before doing the inference process so that we can avoid detecting the rectangle surrounding the match line symbol as walls. Secondly, we will make a list of all failed inference cases and explore the potential improvement. Fourthly, we would like to explore different methods to do “diagonal walls" and “curved walls" detection since the original labeled method would easily make a bounding box of walls covering other objects. The most promising method we would attempt first is an instance segmentation algorithm. Lastly, expanding the automation process to more architectural components, doors and windows, would be a critical contribution for this research and our strategy would be doing multiclass classification and detection, creating precise interface connection among all components.

5 Conclusion

BIM has become an essential methodology in the AEC industry in the past few decades. To facilitate BIM applications, we hope that we could make the BIM model generation effective and precise.

In the paper, we demonstrated that using computer vision techniques to detect the location of walls on the 2D drawings so that we could automatically generate 3D models later on. We achieve good results on the inferred dataset and plan to expand the scale to other components inference and experimental instance segmentation for non-straight walls.

Deep learning related research is very active and advanced, lots of machine learning models are being optimized and well-performed on many projects. Therefore, using those techniques on 2D drawings to 3D model generation could be of practical use in the AEC industry in the near future. These automated systems could reduce budget, improve safety, enhance communication among entities and promote more advanced implementation for researchers and expertise.

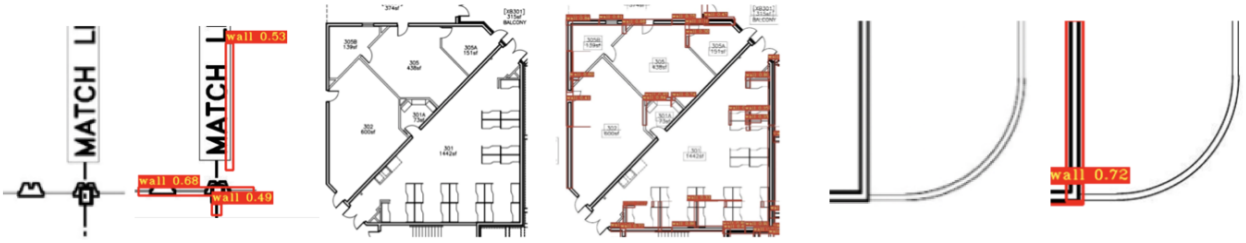

Figure 6. Failure cases: “MATCH LINE” (left), “diagonal wall” (middle) and “curved wall” (right)

References

[1] Michael T. Benefits of Building Information Modeling. On-line: https://www.ny-engineers.com/blog/benefits-of-building-information-modeling, Accessed: 24/01/2022

[2] Xiao Y., Chen S., Ikeda Y. and Hotta K. Automatic recognition and segmentation of architectural elements from 2D drawings by convolutional neural network. In *Proceedings of the 25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*, pages 843–852, Hong Kong, 2020.

[3] Liu C., Wu J., Kohli P. and Furukawa Y. Raster-to-vector: Revisiting floorplan transformation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2195–2203, Venice, Italy, 2017.

[4] Dodge S., Xu J. and Stenger B. Parsing floor plan images. In *Proceedings of the 15th IAPR International Conference on Machine Vision Applications (MVA)*, pages 358–361, Nagoya, Japan, 2017.

[5] Mishra S., Hashmi K., Pagani A., Liwicki M., Stricker D and Afzal M. Towards robust object detection in floor plan images: A data augmentation approach. Applied Sciences. 11(23):11174, 2021.

[6] Zhao Y., Deng X. and Lai H. A Deep Learning-Based Method to Detect Components from Scanned Structural Drawings for Reconstructing 3D Models. Applied Sciences. 10(6):2066, 2020.

[7] Kalervo A., Ylioinas J., Häikiö M., Karhu A. and Kannala J. CubiCasa5k: A dataset and an improved multi-task model for floorplan image analysis. In *Proceedings of the Scandinavian Conference on Image Analysis*, pages 28–40, 2019.

[8] Jang H., Yu K. and Yang J. Indoor Reconstruction from Floorplan images with a Deep Learning Approach. ISPRS International Journal of Geo-Information. 9(2), 2020.

[9] Kippers R., Koeva M., Van Keulen M. and Oude Elberink S. Automatic 3d Building Model Generation Using Deep Learning Methods Based on Cityjson and 2d Floor Plans. In *Proceedings of International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 49–54, 2021.

[10] Seo J., Park H. and Choo S. Inference of drawing elements and space usage on architectural drawings using semantic segmentation. Applied Sciences. 10(20):7347, 2020.

[11] MIT, Computer Science and Artificial Intelligence Laboratory. LabelMe. On-line: http://labelme.csail.mit.edu/Release3.0/, Accessed: 04/11/2021.

[12] Rokas B. PyLessons. On-line: https://pylessons.com/, Accessed: 22/11/2021.