Advances in Artificial Intelligence and Machine Learning; Research 3 (1) 839-852 Received 20-01-2023; Accepted 15-03-2023; Published 21-03-2023

EuclidNet: Deep Visual Reasoning for Constructible Problems in Geometry

Man Fai Wong mfwong29-c@my.cityu.edu.hk
City University of Hong Kong

Xintong Qi xq2224@columbia.edu
Columbia University

Chee Wei Tan cheewei.tan@ntu.edu.sg
Nanyang Technological University

Corresponding Author: Chee Wei Tan

Abstract

In this paper, we present a visual reasoning framework driven by deep learning for solving constructible problems in geometry that is useful for automated geometry theorem proving. Constructible problems in geometry often ask for the sequence of straightedge-and-compass constructions to construct a given goal given some initial setup. Our EuclidNet framework leverages the neural network architecture Mask R-CNN to extract the visual features from the initial setup and goal configuration with extra points of intersection, and then generate possible construction steps as intermediary data models that are used as feedback in the training process for further refinement of the construction step sequence. This process is repeated recursively until either a solution is found, in which case we backtrack the path for a step-by-step construction guide, or the problem is identified as unsolvable. Our EuclidNet framework is validated on the problem set of Euclidea with an average of 75% accuracy without prior knowledge and complex Japanese Sangaku geometry problems, demonstrating its capacity to leverage backtracking for deep visual reasoning of challenging problems.

Keywords: Geometric reasoning, Geometry construction, Deep learning

1. INTRODUCTION

Henri Poincaré once remarked "*Geometry is the art of correct reasoning from incorrectly drawn figures*". Thus, reasoning in geometry often means drawing upon visual figures in a creative manner with *abundant trial and error*. This is true especially for geometric constructible problems that ana-

Citation: Man Fai Wong, et al. EuclidNet: Deep Visual Reasoning for Constructible Problems in Geometry Advances in Artificial Intelligence and Machine Learning. 2023;3(1):52.

lyze the drawing of Euclidean shapes with straightedges and compasses [1, 2]. Every constructible problem requires first a feasibility answer and, if feasible, then the exact sequence of straightedge-and-compass construction steps. For example, though Gauss proved in 1796 the constructibility of the regular seventeen-sided polygon, its explicit construction remains elusive until decades later as shown by J. Erchinger and H. W. Richmond [2]. Gelernter's geometry machine in 1959 would need a diagram computer to generate figures to validate its automated geometry theorem proving [3].

Visualization of figures is undeniably helpful to humans in developing intuition and grasping mathematical concepts in geometry. This process is not only perceptual but includes a visual validation step by step using trial and error to sieve through the visual information [4]. Kim in [5] proposed visual reasoning in geometry by drawing an analogy to human reasoning in identifying visual patterns and analyzing geometric feature information repeatedly. Other works include the diagram reasoning in [6], program synthesis technique in [7] to synthesize geometric constructions, the alignment of visual and textual cues in geometrical diagrams [8] and geometric deep learning [9, 10]. It has become growingly important to understand how to leverage the availability of large data of visual figures to refine and accelerate the trial and error process in *automated visual reasoning*.

In this paper, we propose EuclidNet, a visual reasoning framework that utilizes deep learning for constructible problems in geometry. Deep learning models can be trained on large sets of data to identify patterns where each data instance has a large number of low-level input features. These models can learn to extract higher-level representations of the data through multiple layers of computation, and can be used to guide human intuition to make predictions (e.g., validating mathematical hypotheses in [10]). In particular, EuclidNet leverages the Mask R-CNN to extract visual features from the images and categorize these visual features into points, lines, and circles. Since intersections may also possess significantly important geometric features, we also implement another Mask R-CNN to extract intersections from the image and count them as points. Together, these features form a space where feasible construction steps are derived and form a rooted tree to traverse all possible features. From a variety of features in the tree, EuclidNet selects one geometrical feature to construct and then adds the construction step to the existing construction. Repeating the process, EuclidNet proposes a method of constructing sequences where backtracking occurs when a construction matches the original input image. This allows for a step-by-step reconstruction of a geometrical solution to be derived. This procedure of divide-and-conquer and backtracking is reminiscent of Wang's algorithm for automated theorem proving of propositional logic [11–13], where a statement is decomposed into multiple premises that are considered true, and the automated program exhausts the solution space to determine if the conclusion is valid.

In summary, the contributions of our work are as follows.

- We offer a new perspective on geometric construction using visual reasoning as a comprehensive procedure to discover solutions. Visual reasoning leverages the feature representation of geometric patterns extracted by Mask R-CNN to propose new construction possibilities.
- We introduce backtracking as a means of step-by-step construction solution finding. Once the existing construction is identical to the original input image, a solution is found, and EuclidNet backtracks the construction sequence to reproduce a *proof* of construction.

- We demonstrate the effectiveness of the EuclidNet for solving constructible problems ranging from the classical puzzles found in the popular Euclidea App [14] to more complex ones like the Japanese Sangaku [15–17].

2. RELATED WORKS

2.1 Automated Geometric Reasoning with Formal Reasoning

Automated geometric reasoning refers to the process of deriving theorems from geometric problems with computers, which was first attempted by Gelernter and his collaborators in 1959 [3]. Existing automated geometric reasoning methods primarily utilize formal reasoning, which relies on rules of logic and mathematics to conduct reasoning. These methods can be further categorized into synthetic and algebraic approaches. For synthetic reasoning, geometric definitions and theorems are built inside machines, which then incorporate search techniques, often exhaustive, to find the solution [7]. Examples of the synthetic approach can be found in Inter-GPS [18] and GEOS [8], which are solvers with built-in Euclidean formulas. The algebraic approach involves algebraic operations such as Wenjun Wu's method [19] that decomposes problems based on the well-ordering principle and successive pseudo-reduction. However, the algebraic approach has many limitations, and no geometric construction solver has adopted this method thus far.

2.2 Automated Geometric Reasoning on Constructions

An early attempt to perform geometric reasoning on constructions with images was made by Gao [20] to extract information from the diagram and produce a construction sequence. A more recent image-based geometric construction solver is built on top of a Mask R-CNN to find geometric constructions with straightedges and compasses [21]. The solver uses a Mask R-CNN to generate geometric images that however do not yield solutions that conform to formal reasoning. To be more specific, their approach trains deep learning models based on images generated by the Euclidea App with variations such as rotation and translation to be used as the training dataset. However, this image-based solver yields discriminative learning models that though perform effectively are limited by the training data. The possibility to integrate generative models and a systematic form of image-based reasoning can lead to a new approach to explore constructible problems in geometry. The work in this paper has been presented in part at the 2nd MATH-AI Workshop at NeurIPS'22.

3. METHODOLOGIES

EuclidNet is a visual reasoning system that uses lines and circles to generate new points by deep learning on a visual image, and then employs a backtracking algorithm to exhaustively explore all feasible possible moves that follow the rules and solutions to the constructible problems.

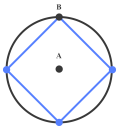
3.1 Elementary Euclidean Constructions Procedure

In a Euclidean plane, points, lines and circles are considered the fundamental constructing elements, which are also known as primitives [1, 2]. When we solve constructible problems in geometry, it is assumed that specific points are known a priori in the infinite Euclidean plane [22]. With line and circle tools, only limited moves can be made with the existing points in the Euclidean plane:

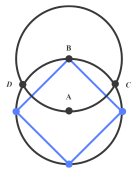
1. Given two non-identical points $A(x_1, y_1)$ and $B(x_2, y_2)$, we can draw the unique straight line $L = AA$ with the straightedge containing both points. The ray-line will be projected to the canvas boundary by the slope $m = \frac{y_2 - y_1}{x_2 - x_1}$ and y-intercept $b = y_1 - mx_1$ or $b = y_2 - mx_2$.
2. Given two points $C(h, k)$, M and $|CM| > 0$, we can use the compass to draw a unique circle $c = \{C; |CM|\}$ with C being the center, $|CM|$ being the radii, and M on the circle.

Our geometric construction environment uses only basic tools such as point, line, and circle, yet it can construct advanced tools like perpendicular bisector and angle bisector. FIGURE 1, illustrates a solution to a problem from Euclidea (Alpha-7) with the initial configuration in black and the goal configuration in blue. One can find a sequence of steps using a ruler and compass to move from an existing configuration to a given goal configuration. We can define new points in the plane, such as intersections and points from a ray line from the previous moves.

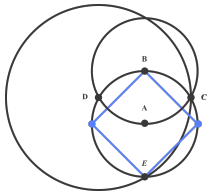
There are three types of intersections we consider: *line-line*, *line-circle*, and *circle-circle*. For the two non-parallels, there is not necessarily an intersection point for a line segment, but an *line-line intersection* point must exist with their ray. Secondly, we consider the *line-circle intersection*. There are three ways a line and a circle can be associated, i.e., the line cuts the circle at two distinct points, the line is tangent to the circle, or the line misses the circle. Finally, we define the cases on *circle-circle intersection*. If the sum of the radii and the distance between the centers are equal, then the circles touch externally. Also, if the difference between the radii and the distance between the centers are equal, then the circles touch internally.



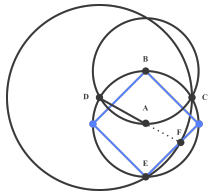
Input: Inscribed Square: Given a circle with center A and a point B on circle, inscribe a square in the circle.



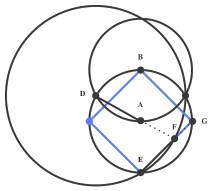
Step 1: Draw a circle from A as a center to B. Two points C and D from circle-circle intersection will be obtained.



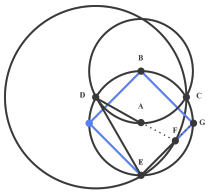
Step 2: Draw a circle from D as a center to C. A point E from circle-circle intersection will be obtained.



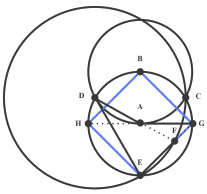
Step 3: Draw a line from D to A, a point F will be obtained from a ray of DA.



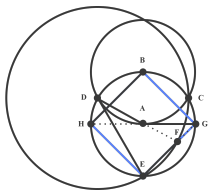
Step 4: Draw a line from E to F, a point G will be obtained from a ray of EF.



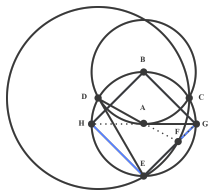
Step 5: Draw a line from D to E.



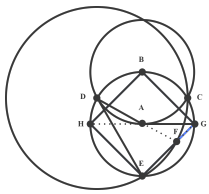
Step 6: Draw a line from G to A, a point H will be obtained from a ray AG.



Step 7: Draw a line from H to B.



Step 8: Draw a line from B to G.



Step 9: Draw a line from H to E and the construction is completed.

Figure 1: Illustration of construction on the inscribed square in a circle using straightedges and compasses on the geometric construction environment of EuclidNet.

3.2 Deep Visual Reasoning

Although visual reasoning is second nature to humans given the visual inputs [5], it is not trivial for computers to emulate this. Computers need to first sieve through geometric patterns that are visually depicted, and then comprehend logical relationships between patterns (e.g., isometries) in order to reason. Our approach is to employ deep learning in [23] for geometric primitive extraction as well as image segmentation for pattern-seeking so as to enable data-driven pattern recognition for geometrical diagrams. In some sense, the interpretability of the trained machine learning model for automated visual reasoning is to turn geometrical diagrams into *proof without words* for human understanding. We implement EuclidNet by deep visual reasoning functionality with Mask R-CNN.

3.3 Backtracking Algorithm

EuclidNet performs an exhaustive search on the space of all possible moves to enumerate all possible compositions of tools and verifies if any of the constructions match the given target construction. In an exhaustive search, we develop a backtracking algorithm to construct the geometries recursively to build the solution incrementally, one possible move at a time, and we remove those constructions that fail immediately. The final solution is derived from tracing backward and assembling the steps together to form an integrated whole. This backtracking procedure is described in FIGURE 2. The

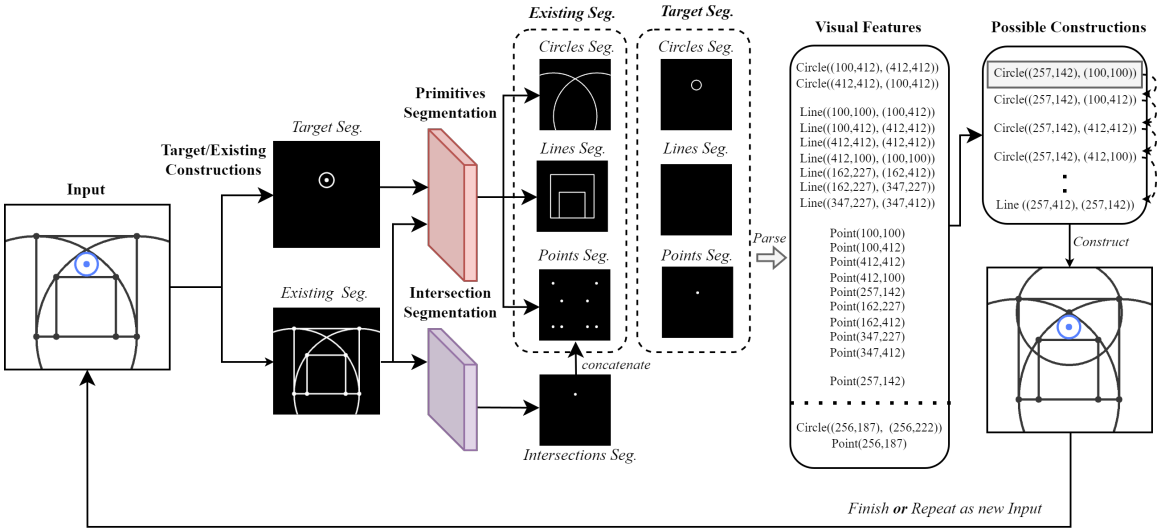


Figure 2: The architecture of EuclidNet. It consists of two pre-trained models for segmenting and locating geometric primitives and intersections.

search stops when a solution is found or there are no more possible constructions. The search algorithm takes an image of a constructible problem and an empty sequence of images as the initial input. The primitives and intersections information will be extracted by the pre-trained segmentation models for each input image of the search simultaneously. Alternatively, the information can be carried forward to the next depth and skip the extraction during the search. Since the search is implemented by backtracking, it may be expensive to carry forward with all the information when the problem is too complex. The search will stop if a solution is found, and the sequence of images will then be saved. The implementation of the algorithm is defined as follows:

3.4 Network Architecture

We implement our EuclidNet for the segmentation and localization of primitives and intersections on top of Mask R-CNN [24]. The input RGB image will be divided into two images for each channel and carried forward to the network separately. The detailed architecture is illustrated in FIGURE 3.

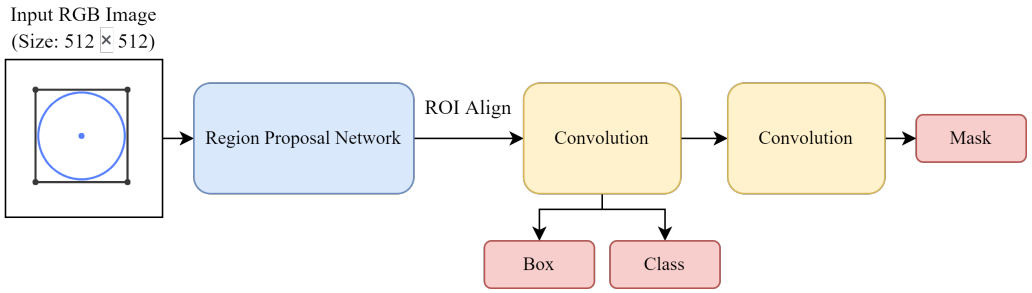


Figure 3: Illustration of Mask R-CNN architecutre in our framework.

Algorithm 1 (**DVRB**) **D**eep **V**isual **R**easoning with **B**acktracking

Input: Image I_{curr}, Depth D_{curr}, Sequence of Images $\boldsymbol{S}$
Output: Sequence of Images $\boldsymbol{S}$
Variables: Possible Moves $\boldsymbol{M}$, Points $\boldsymbol{P}$, Lines $\boldsymbol{L}$, Circles $\boldsymbol{C}$
Parameters: Goal Image I_{goal}, Maximum Depth D_{max}
Models: Primitives Seg_p , Intersections Seg_i ▷ Pre-trained segmentation models

```
if I_curr = I_goal then                                   ▷ Check I_curr is solved
    save(S)                                               ▷ Save the solution
    return
else
    if D_curr = D_max then
        return                                            ▷ Reach the maximum depth
    else
        P, L, C ← Seg_p.extract(I_curr)                   ▷ Extract primitives
        P ← P + Seg_i.extract(I_curr)                     ▷ Extract intersections for existing points
        M ← construct(P)                                 ▷ Construct all moves defined in section 2.1
        M ← M.remove(L, C)                               ▷ Remove existing lines and circles from possible moves
        while M not empty do
            m ← M_0                                       ▷ Pick the first possible move in M
            S ← I_curr ⊕ m                                ▷ Add the concatenation of a new move and current image
            DVRB(I_curr ⊕ m, D_curr + 1, S)               ▷ Substitute new moves to next depth
            S.remove(I_curr ⊕ m)                          ▷ Remove a tested construction
            M.remove(m)                                   ▷ Remove a tested move
```

3.5 Diagram Element Segmentation and Localization

EuclidNet is implemented for segmenting and locating primitives and intersections using Mask R-CNN [24] with Resnet-50 and Resnet-101 backbones. The Mask R-CNN generates regional proposals, classifies foreground/background anchors, and uses RoI align to generate fixed-size feature maps. Three tasks are trained together: classification, box regression, and mask regression, with a total loss defined as the sum of classification loss (L_{cls}), bounding-box loss (L_{box}), and average binary cross-entropy mask loss (L_{mask}). The total training loss can be defined as:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask}, \tag{1}$$

where L_{cls} is the loss of the classification, L_{box} is the bounding-box loss that is identified as those defined in [25], and L_{mask} is the average binary cross-entropy loss including solely the k-th mask if the region is associated with the ground truth class k from [24]. We have employed a primitive segmentation model and an intersection segmentation model to extract geometric information and emphasize intersections for improved visual reasoning.

4. EXPERIMENT

We have created a dataset for primitives and intersections extraction by adding different primitives and intersections to images for experiments. To measure model performance, we use the mean

average precision (mAP) as the evaluation metric. We have evaluated the performance of EuclidNet on Euclidea App puzzles [14] and found it to be effective. It can recognize patterns like isometries and solve complex problems like Japanese Sangaku problems. An example of EuclidNet's problem-solving ability is shown in FIGURE 4. Moreover, EuclidNet demonstrates effectiveness in solving complex geometric construction questions such as the Japanese Sangaku problems [15, 17]. The source code and data are available at `https://github.com/EuclidNet/EuclidNet`.

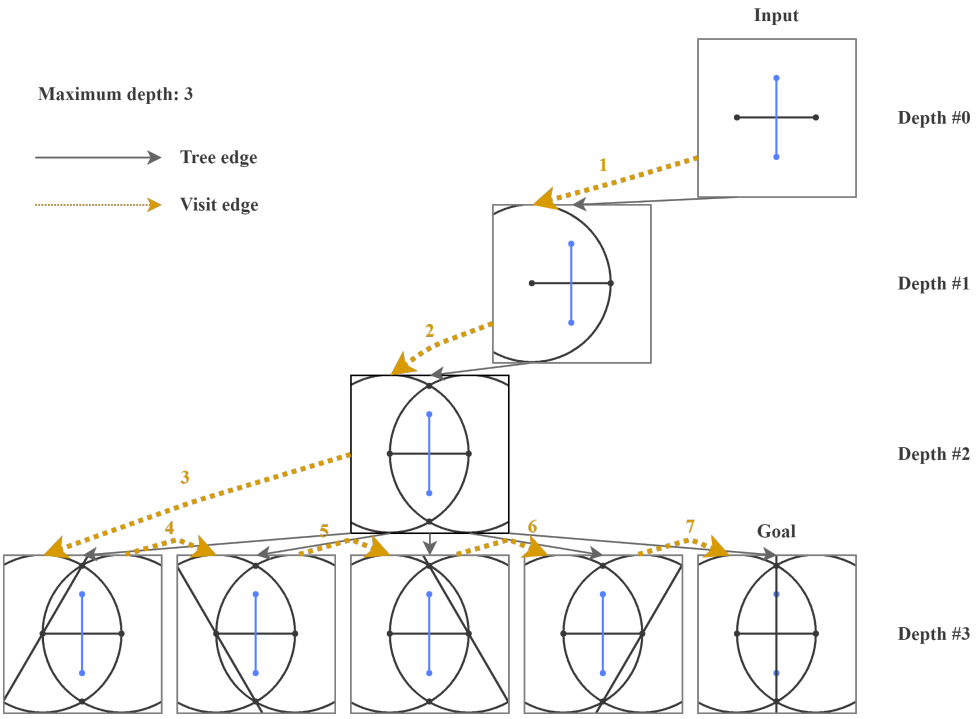


Figure 4: Illustration of the construction of a perpendicular bisector on a line. The problem starts with two points on a line (in black) and requires finding the perpendicular bisector (in blue). If the search reaches the maximum depth, the search process will start over.

4.1 Experimental Setup

We evaluate the performance of our models in terms of the mean average precision (mAP) for both primitives and intersections. We experiment with ResNet-50 and ResNet-101 as backbones for Mask R-CNN because they balance accuracy and computational efficiency. ResNet101 is more accurate than ResNet50 but also more computationally expensive. Compared to other models for Mask R-CNN backbones, InceptionV3 is more computationally efficient than ResNet50, but it may not perform as well on our task. MobileNet is designed to be even more computationally efficient than InceptionV3, but it may sacrifice some accuracy for speed. To train segmentation models, we set the SGD optimizer with a learning rate of 0.001 and the minimum detection confidence as 0.7. The training is launched on a single NVIDIA 2080Ti GPU (11GB) with a batch size of 16. Other parameters follow the default configuration in [24]. The models are trained on 1000 simulating images with primitives and intersections separately on 200 epochs and 1000 steps per epoch.

4.2 Experimental Results

TABLE 1, shows the performance of the trained Mask R-CNN with different backbones. TABLE 2, shows the performance in solving geometric problems of the Euclidea App puzzles [14]. The result shows that our models can detect the targets with mAP over 90%. Euclidea does not yet have puzzles involving the concept of area, but it works well on the other constructible problems.

Table 1: Performance comparison: Mask r-cnn with backbones resnet-50 and resnet-101 on the primitives and intersection segmentation.

Backbone	**Primitives**				**Intersections**			
	mAP	AP_{50}	AP_{75}	AP_{90}	mAP	AP_{50}	AP_{75}	AP_{90}
ResNet-50	0.936	0.949	0.945	0.868	0.926	0.911	0.908	0.901
ResNet-101	0.938	0.951	0.948	0.866	0.928	0.914	0.912	0.906

Table 2: Evaluation result on the Euclidea app puzzles

	Alpha	**Beta**	**Gamma**	**Delta**	**Epsilon**	**Zeta**
Accuracy	0.857	0.900	0.778	0.636	0.727	0.636

4.3 Illustrative Examples

4.3.1 Euclidea app puzzle (*Alpha-4*): Inscribed circle

Given a triangle, an inscribed circle is the largest circle contained within the triangle. The inscribed circle will touch each of the three sides of the triangle at exactly one point. The computational process is illustrated in FIGURE 5.

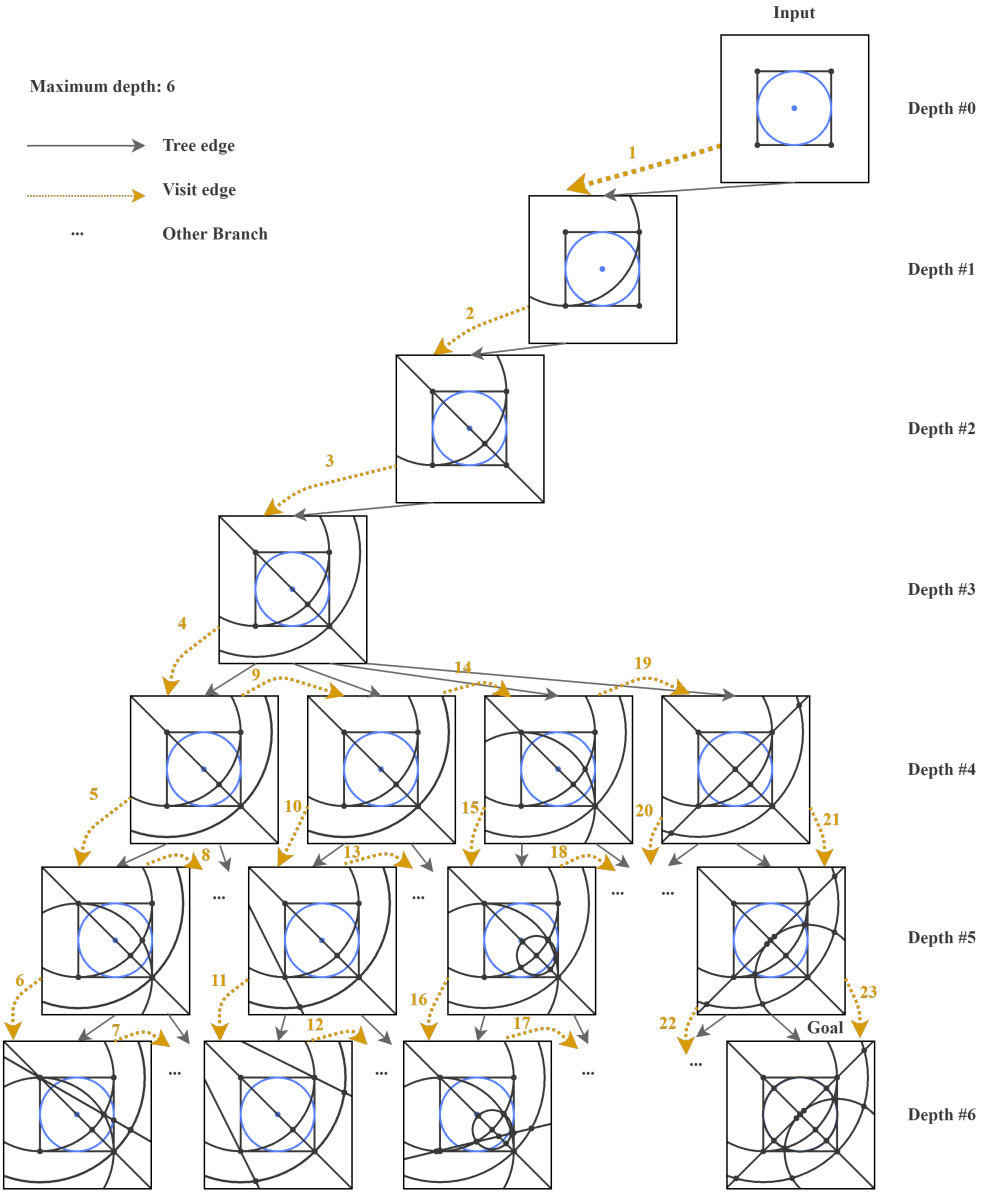


Figure 5: Illustration of the construction of a circle inscribed in the square (Euclidea *Alpha-4*). The maximum depth in this example is 6.

4.3.2 Euclidea app puzzle (*Gamma-5*): Circle through a point tangent to line

A circle through a point tangent to the line is to construct a circle through the arbitrary point that is tangent to the given line at the point on the line which is illustrated in FIGURE 6.

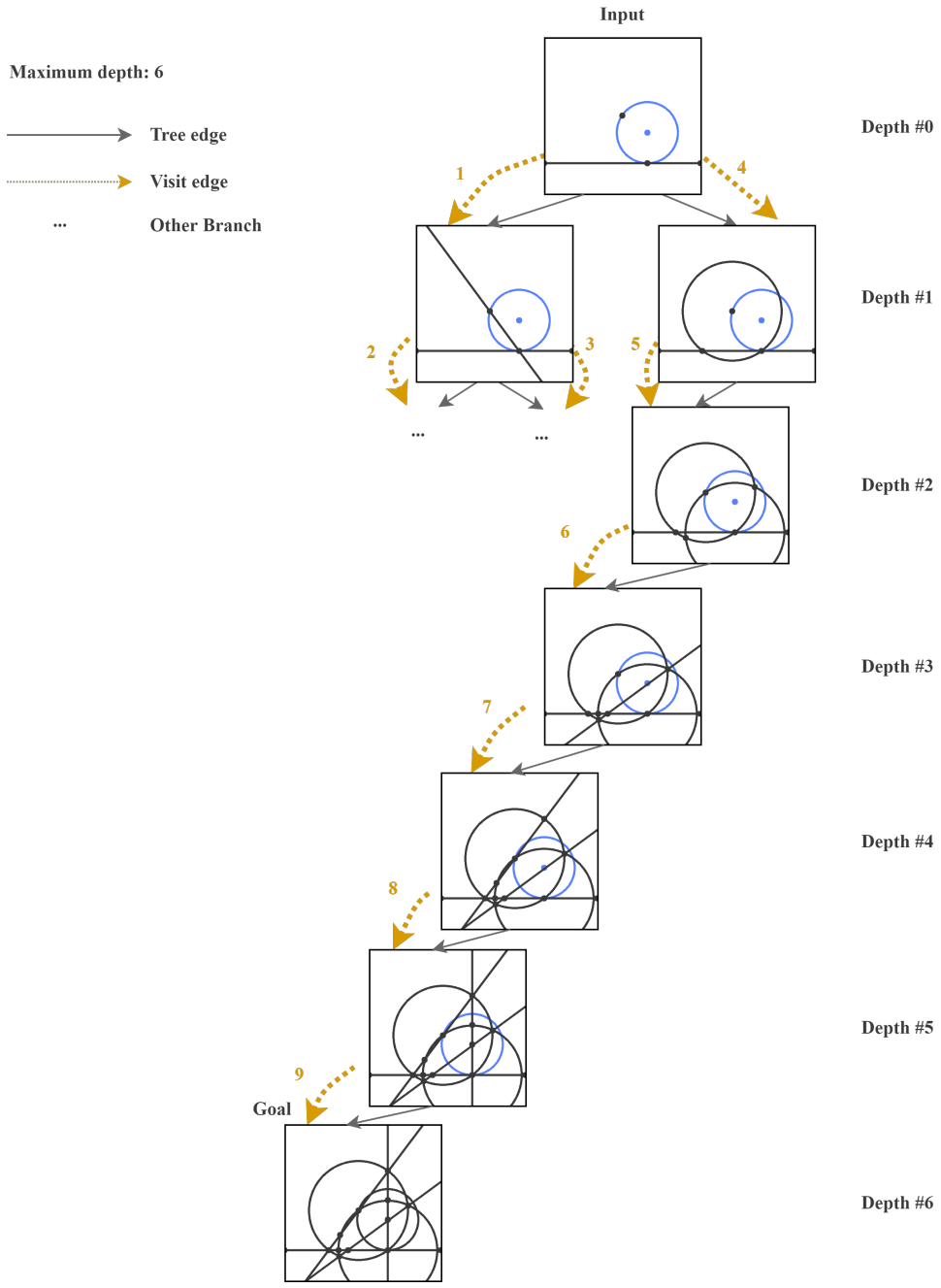


Figure 6: Illustration of the construction of a circle through a point tangent to line. The maximum depth in this example is 6.

4.3.3 Sangaku: Sangaku with Versines

Sangaku with Versines was created in 1825 in the Tokyo prefecture, and connects the rarely-used versine quantities with the distance from a vertex to the incircle [16, 17]. The computational process is illustrated in FIGURE 7.

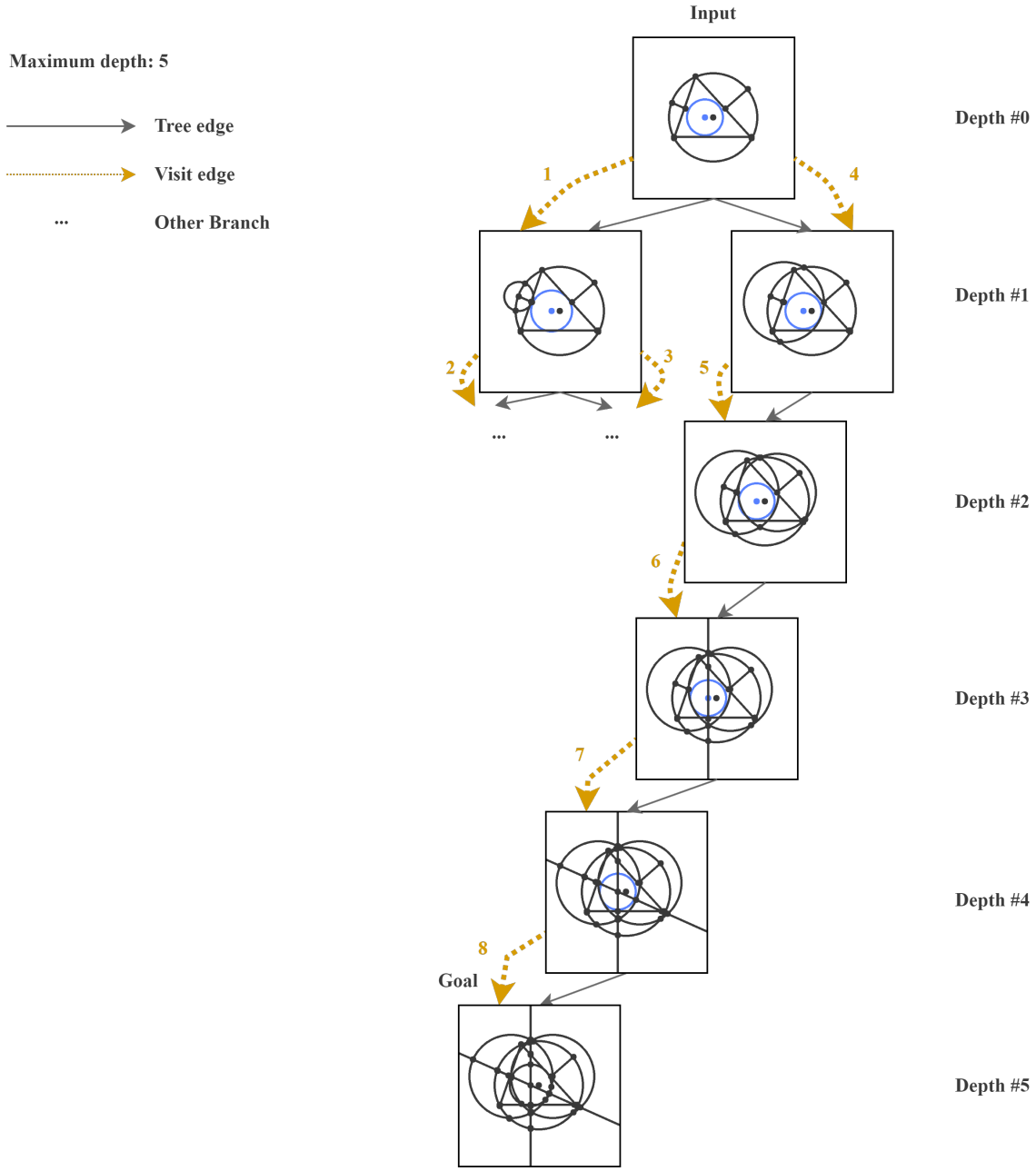


Figure 7: Illustration of the construction of Sangaku with versines. The maximum depth in this example is 5.

4.3.4 Sangaku: Square and circle in a gothic cupola

Square and Circle in a Gothic Cupola from [16] is a Sangaku with two-quarter circles inscribed in a square form a gothic cupola, with a small circle standing on top of a bigger circle [17]. The computational process is illustrated in FIGURE 8 .

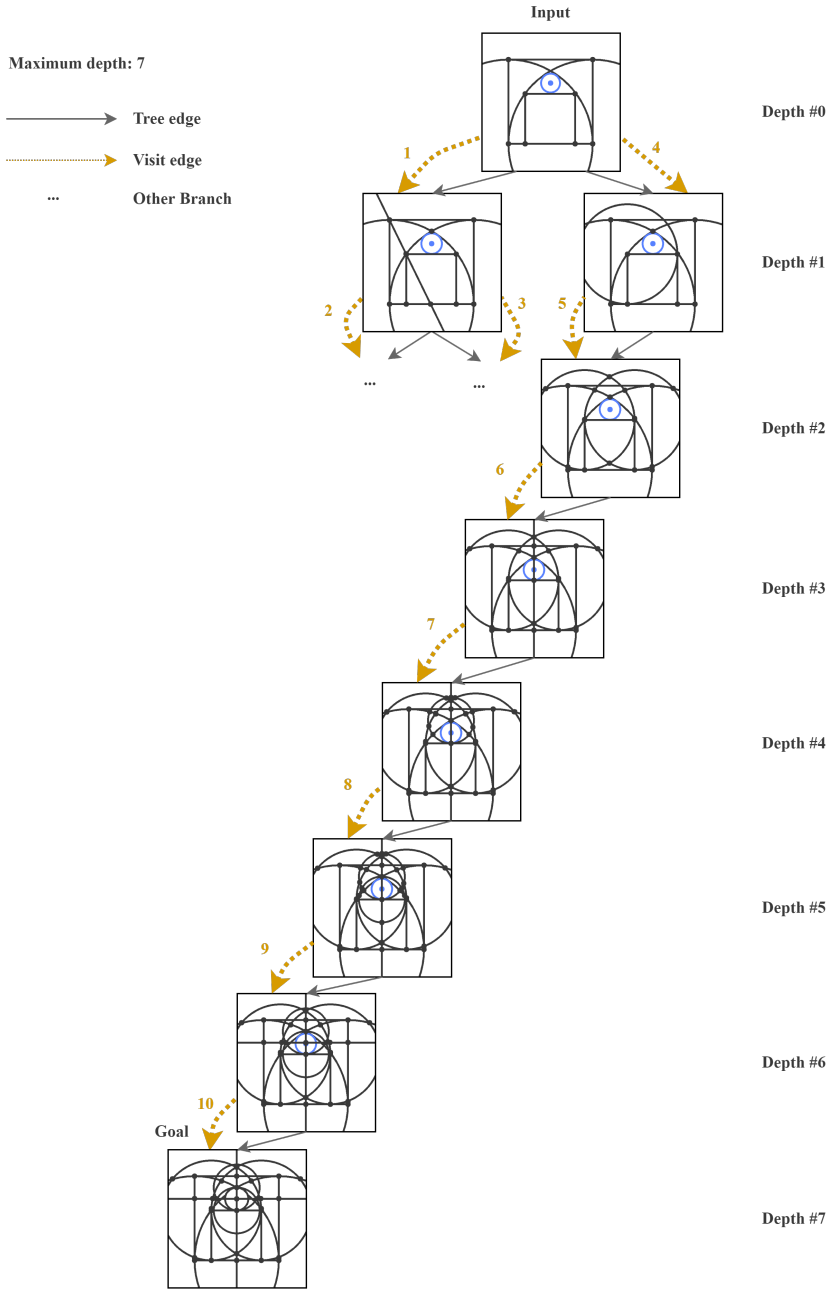


Figure 8: Illustration of the construction of a square and circle in a Gothic Cupola. The maximum depth in this example is 7.

5. CONCLUSION

In this paper, we present EuclidNet, a novel visual reasoning framework driven by deep learning for solving constructible problems in geometry. The framework localizes geometric primitives using Mask R-CNN models and identifies new intersections from previous moves to discover new constructions. To find the solution, EuclidNet employs a backtracking algorithm to search for possible constructions with relatively low computational complexity. To validate EuclidNet's effectiveness to explore the solution space of constructible problems and its efficacy at reasoning 'constructions', we have conducted various experiments on numerous challenging constructible problems, including Japanese Sangaku geometry. While our results demonstrate visual reasoning by deep learning, there are some limitations. For more challenging geometry problems (or those with more geometrical objects), backtracking can be time-consuming due to the need to explore a large number of possibilities before finding a solution. It can be useful to incorporate the Visual Turing Test in [26] with visual deep reasoning to prune the search space. Learning a fine-grained search space using the Visual Turing Test can also lead to improved training data that contributes towards model accuracy. As future work, it can be interesting to extend EuclidNet with neural-symbolic artificial intelligence or few-shot learning for geometry constructible problems arranged in increasing order of difficulty.

6. ACKNOWLEDGEMENT

This research was supported in part by the Ministry of Education, Singapore, under its Academic Research Fund (No. 022307). C. W. Tan acknowledges helpful discussions with Mikhail Koroteev.

References

[1] Hilbert D. The Foundations of Geometry. Open Court Publications Company. 1902.

[2] Pedoe D. Geometry, a Comprehensive Course. Dover books on mathematics. 1988.

[3] Gelernter HL, Rochester N. Intelligent Behavior in Problem-Solving Machines. IBM J Res Dev. 1958;2: 336-45.

[4] Dreyfus T. On the Status of Visual Reasoning in Mathematics and Mathematics Education. In: Proceedings of the 15th conf. of the int group for the psychology of mathematics education. 1991.

[5] Kim MY. Visual Reasoning in Geometry Theorem Proving. In: Proceedings of the 11th international joint conference on artificial intelligence. 1989.

[6] Koedinger KR, Anderson JR. Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry. Cogn Sci. 1990;14:511-550.

[7] Gulwani S, Korthikanti VA, Tiwari A. Synthesizing Geometry Constructions. SIGPLAN Not. 2011;46:50-61.

[8] Seo M, Hajishirzi H, Farhadi A, Etzioni O, Malcolm C. Solving Geometry Problems: Combining Text and Diagram Interpretation. In: Proceedings of the 2015 conference on empirical methods in natural language processing; 2015:1466-1476.

[9] https://arxiv.org/abs/2104.13478

[10] Davies A, Veličković P, Buesing L, Blackwell S, Zheng D, et al. Advancing Mathematics by Guiding Human Intuition With AI. Nature. 2021;600:70-74.

[11] Wang H. Computer Theorem Proving and Artificial Intelligence. In: Automated theorem proving: after. Springer; 1984;29:49-70.

[12] H. Wang. Toward Mechanical Mathematics. IBM J Res Dev. 1960;4:2-22.

[13] Wang H. Proving Theorems by Pattern Recognition I. Commun Assoc Comput Mach. 1960;3:220-234.

[14] https://www.euclidea.xyz

[15] Fukagawa H, Pedoe D. Japanese Temple Geometry Problems. Charles Babbage Research Centre. 1989.

[16] Fukagawa H, Rothman T. Sacred Mathematics: Japanese Temple Geometry. Princeton University Press. 2008.

[17] https://www.cut-the-knot.org/pythagoras/Sangaku.shtml

[18] Lu P, Gong R, Jiang S, Qiu L, Huang S, et al. Inter-GPS: Interpretable Geometry Problem Solving With Formal Language and Symbolic Reasoning. In: The 59th Annual Meeting of the Association for Computational Linguistics. 2021.

[19] Tun WD. On the Decision Problem and the Mechanization of Theorem-Proving in Elementary Geometry. Sci Sin. 1978;21:159-172.

[20] Gao XS, Chou SC. Solving Geometric Constraint Systems. II. A Symbolic Approach and Decision of RC-Constructibility. Comput Aid Des. 1998;30:115-122.

[21] Macke J, Sedlar J, Olsak M, Urban J, Sivic J. Learning to Solve Geometric Construction Problems From Images. In: International Conference on Intelligent Computer Mathematics. Springer; 2021:167-184.

[22] Geretschläger R. Euclidean Constructions and the Geometry of Origami. Math Mag. 1995;68:357-371.

[23] LeCun Y, Bengio Y, Hinton G. Deep Learning. Nature. 2015;521:436-444.

[24] He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. Proceedings of the IEEE international conference on computer vision. 2017:2961-2969.

[25] Girshick R. Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.

[26] Geman D, Geman S, Hallonquist N, Younes L. Visual Turing Test for Computer Vision Systems. Proc Natl Acad Sci U S A. 2015;112:3618-3623.