Slow Perception: Let's Perceive Geometric Figures Step-by-step

Haoran Wei[1]*, Youyang Yin[2]*, Yumeng Li[2], Jia Wang[1],
Liang Zhao[1], Jianjian Sun[1], Zheng Ge[1], Xiangyu Zhang[1]
[1]Stepfun   [2]Beihang University
https://github.com/Ucas-HaoranWei/Slow-Perception

Abstract

Recently, "visual o1" began to enter people's vision, with expectations that this slow-thinking design can solve visual reasoning tasks, especially geometric math problems. However, the reality is that current LVLMs (Large Vision Language Models) can hardly even accurately copy a geometric figure, let alone truly understand the complex inherent logic and spatial relationships within geometric shapes. We believe accurate copying (strong perception) is the first step to visual o1. Accordingly, we introduce the concept of "slow perception" (SP), which guides the model to gradually perceive basic point-line combinations, as our humans, reconstruct complex geometric structures progressively. There are two-fold stages in SP: a) perception decomposition. Perception is not instantaneous. In this stage, complex geometric figures are broken down into basic simple units to unify geometry representation. b) perception flow, which acknowledges that accurately tracing a line is not an easy task. This stage aims to avoid "long visual jumps" in regressing line segments by using a proposed "perceptual ruler" to trace each line stroke-by-stroke. Surprisingly, such a human-like perception manner enjoys an inference time scaling law—the slower, the better. Researchers strive to speed up the model's perception in the past, but we slow it down again, allowing the model to read the image step-by-step and carefully.

1. Introduction

Geometric figure parsing, entailing the conversion of geometric shapes in 2D images into editable, is a significant task in computer vision, which enjoys promising academic and industrial values. In the realm of research, geometric figure perception has the potential to prompt the mathematical visual reasoning field [39, 6, 19, 28]. Meanwhile, in applied domains, it also holds landing prospects in education, architecture, and other fields. However, geometry parsing
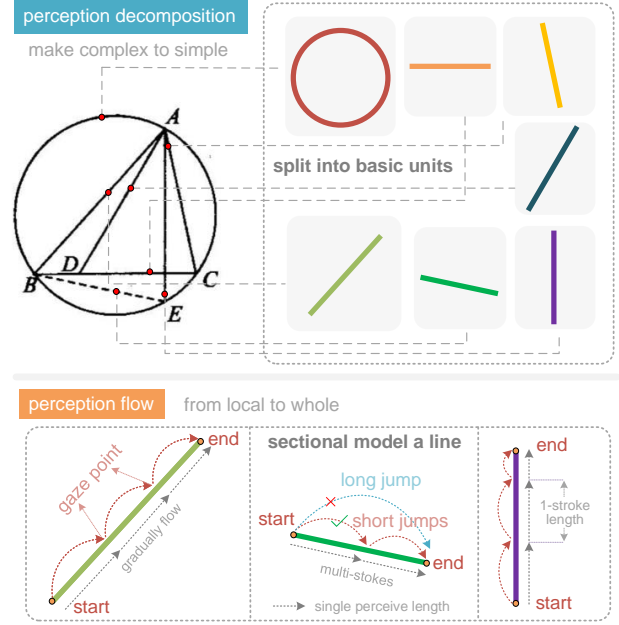

Figure 1. Slow perception enjoys two stages: 1) Perception decomposition. A geometric shape is decomposed into basic visual units, such as circles and line segments, thereby unifying the fundamental representational form of diverse geometric figures. 2) Perception flow. Using the same modeling approach (predicting the endpoint based on the starting point) for line segments of different lengths is unreasonable. We employ a sectional copying method to express each line segment with a perceptual ruler.

is not easy due to the spatial relationships and dependencies among geometric units. To our knowledge, there are no effective solutions, pretrain data, or valid benchmarks so far, which further hinders the development of this field.

Over the last few years, when detection algorithms were particularly popular [26, 25, 14, 12, 35, 31, 30], utilizing detection models for geometric parsing is a considered approach [11]. However, different from natural objects [13], geometric shapes inherently possess element relations. For example, in $\angle ABC$ as shown in Figure 1, the sides AB
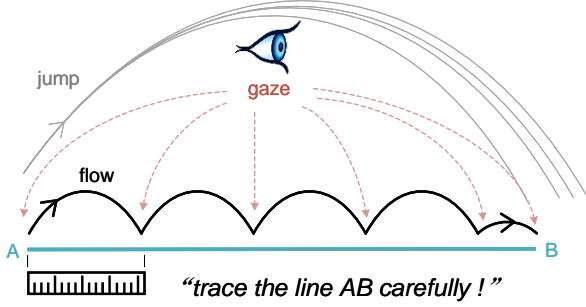
---
*Equal contribution

Figure 2. When humans trace a line, it is typically a slow perception process. Rather than sketching the line, especially a long line, in one stroke (long range "jump"), humans commonly draw a line with "multiple short strokes" for high precision. Our "slow perception" algorithm is designed based on this to mimic the gradual human process of discerning geometric figures.

and BC converge at the common vertex B, whereas object detection methods predict object targets independently (in parallel). As a result, the output results of line AB and BC may yield inconsistent coordinates for point B. This decoupling prediction manner is why traditional detection struggles with geometric figure parsing tasks.

In the past two years, LVLMs [21, 2, 32, 33, 40] have demonstrated exceptional capabilities in image description [13] and visual question-answering [27, 20] tasks. More importantly, for the geometry parsing task, the next token prediction modeling approach ensures that subsequent points can reference the coordinates of previously generated ones, thereby guaranteeing the closure of the output geometric shape, leading us to hope LVLMs could solve geometry parsing problems. However, when we attempt to use state-of-the-art models [3, 34, 2, 21, 16], to generate code for de-rendering geometric shapes, we find all of them, even like GPT-4o [21] and Claude3.5 [1], fail to demonstrate this capability. For us humans, copy a geometric shape seems to be a straightforward task that even an elementary school student can perform well with just a ruler. This compels us to pay close attention to this task. A natural question arises: *where do current LVLM modeling paradigms fall short?*

Imagine if we are to trace a geometric shape manually. We never accomplish this in one fell swoop (in one stoke). Instead, the typical methodology involves: 1) disassembling complex geometric shapes into small units (aiming to "make complex to simple"); and 2) drawing each visual part stroke-by-stroke ("from local to a whole"), as shown in Figure 2. We claim it is the answer to the above question, and following this, we propose the "slow perception" (SP) concept to guide the model to do such a task as humans.

Specifically, regardless of how complex a geometric shape may be, it can always be decomposed into the most basic combinations of points and lines. This allows for a

unified geometric representation of all shapes. For instance, don't need to care about what polygon a geometric shape is, the model only needs to predict each line segments that compose it in a certain order. This we call the perception decomposition (1-order slow-down). However, modeling a line segment is not as simple as just considering it as paired endpoints. This definition faces two problems: 1) The number of tokens that represent the line segment is fewer than those representing the endpoints, leading to insufficient optimization of the line segment (the relation between points). This can result in accurate point prediction but chaotic connections between points. 2) The computational cost for predicting long and short lines is the same, which contradicts our intuitive perception. Inspired by the ruler tool and eye movement process humans use when copying geometric line segments (Figure 2), we propose the perception flow (2-order perception slow-down), which employs a segmented tracing method to represent each line. Specifically, each line segment can be represented as: "start point → gaze point 1 → gaze point 2 → ⋯ → gaze point n → endpoint". The value of n is related to both the length of the target line segment and the preset "perceptual ruler".

Most importantly, along with the concept and modeling approach of slow perception, we provide a method for rendering geometric shapes to scale up the dataset. We have constructed a total of 200,000 synthetic data samples for training a model. Additionally, we collect manually annotated 480 real-world geometric figures from middle school exam paper scenarios, with 120 for validation and 360 as a test set. We will open-source all data and codebase to promote community development in such a field.

Experimentally, slow perception can improve the F1-score by 6%. We also discover two interesting conclusions: 1) The perception flow method for line prediction consistently improves accuracy. Even when the perceptual ruler is set to a relatively large value, it can solidly enhance performance. 2) Slow perception exhibits an **inference time scaling law**: as the perceptual ruler decreases in length, the computational cost for predicting each line segment increases, leading to longer inference times, which gradually improves the geometric parsing performance.

In summary, geometric shapes are human abstractions of natural vision objects. Thereby, we believe our findings in the geometry parsing task will provide insights for other research areas of computer vision as well.

2. Related Works

2.1. Object Detection as Vision Perception

Object detection [9, 8, 26, 24, 25] is one of the hottest research topics in computer vision, which can be broadly categorized into two-stage [26] and one-stage [24]. Previously, it was believed that two-stage methods offered higher
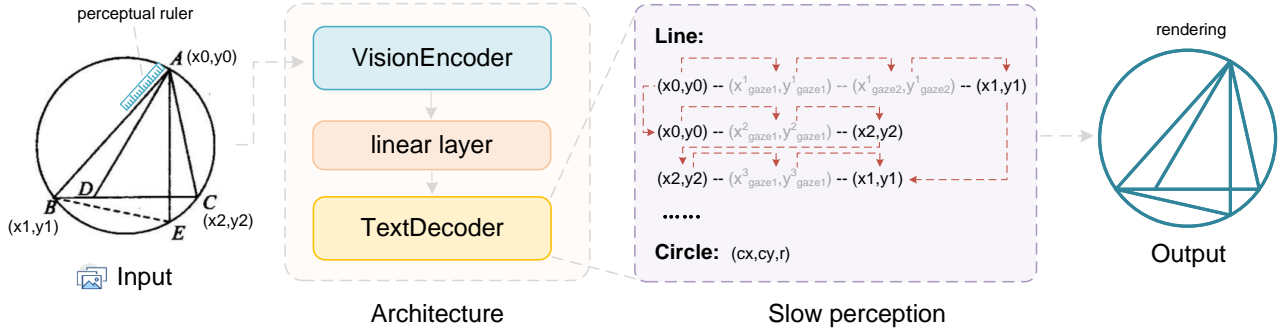
Figure 3. The framework of slow perception. Our approach is adaptable to the most popular LVLM frameworks. According to the next-token serialized prediction, predicted subsequent geometric points can reference the coordinates of preceding points to achieve closed shapes more easily. We establish a perceptual ruler as the upper limit for single-step distance prediction.

accuracy while one-stage methods were faster. Later, with the further development of foundational models, algorithmic engineering, and transformer [4] networks, one-stage models have become both powerful and efficient. In recent years, the prevailing trend in detection algorithms seems to have been dominated by the one-stage type.

For the geometric parsing task, using object detection algorithms does not seem to make sense. This is because independently detecting each geometric visual component cannot guarantee the whole geometric closure. For instance, point A often serves as the endpoint of multiple different line segments, and the parallel prediction of each line cannot ensure the consistency of this point, even if the error is minimal. Furthermore, from RCNN [9] to Faster RCNN [26], and then to the YOLO series [24, 25], this is a trend towards increasingly faster perception. However, we can't help but question: is faster perception always better? is perception purely an optimization problem? do we annotate objects in dense areas, small objects, or extremely large objects at the same speed?

2.2. LVLM for Vision Perception

Recently, research on large vision-language models (LVLMs) [16, 2, 37, 7] has been on the rise, and these models have demonstrated state-of-the-art performance in various visual perception tasks, such as OCR [27, 34, 15, 5] and grounding [40, 38]. After more than a year of development, the framework of these LVLMs has become quite convergent. Specifically, new models often adopt an "encoder-perceiver-decoder" architecture and utilize a training approach similar to large language models (LLMs), primarily involving pretraining followed by supervised fine-tuning (SFT). It is worth noting that the powerful visual knowledge (open-set universal object recognition capability) of LVLMs has also left a deep impression on people, leading us to have very high expectations for LVLMs.

However, some works like BlindTest [23] show that

LVLMs don't seem to understand images truly; in other words, the models look at images too superficially. This cursory glance manner of reading makes it difficult to capture details, logic, and spatial relationships within the image. Some works have attempted to enhance VLM capabilities using a chain of thought [36] approach. What's puzzling is: does perceiving an object multiple times; or only reading an image carefully, require thinking?

3. Methodology

3.1. Architecture

As shown in Figure 3, we chose the classic LVLM framework for experiments to verify the efficiency of slow perception. It usually consists of a vision encoder preceding an LLM decoder, with a simple linear layer in between for channel mapping. Specifically, we use GOT-OCR2.0 [34] as the primary experimental model due to its iterative efficiency. Additionally, we utilize other classic LVLMs, e.g., Qwen2-VL [29] and Vary [33], to further validate the effectiveness of our slow perception.

3.2. Data Engine

We render 200k synthetic geometric images as the train data, wherein Matplotlib is employed as the rendering engine. We stochastically vary multiple parameters to ensure data heterogeneity, including line thickness, line style (solid or dashed), and image resolution (DPI). In total, 150k images are generated with DPI values randomly distributed between 36 and 300, while the remaining 50k are uniformly set to 96 DPI, reflecting a commonly used resolution in practical applications. For the corpus of point-line locations and relationships that make up geometry, we devise the following generation procedure:

1) Selection of substrate. We select the most common quadrilaterals as the rendering base, including squares, rectangles, parallelograms, rhombuses, trapezoids, isosceles

trapezoids, right trapezoids, and other uncommon arbitrary quadrilaterals.

2) Addition and deletion of points. Based on base quadrilaterals, we randomly delete 0-1 points or add 1-6 points to augment the polygon diversity. For extra generated points, we mainly take them on the sides or side extensions of the base figure, with increased probability weights of selecting the mid- or trisection points.

3) Generation circular and text. With a predetermined probability, we add inscribed and circumscribed circles for the base quadrilaterals. Besides, text labels ("A" to "Z") are generated at vertex positions with a certain probability. Although these features are not used in slow perception, their inclusion will enhance the resemblance of the rendered data to real-world geometric shapes.

The entire rendering process aforementioned can be represented as follows:

$$\mathbf{G} = \Phi(\Psi(q, \mathbf{P}), \delta, \mathbf{A}, \omega, \rho_c, \rho_t, \mathcal{T}) \qquad (1)$$

where $\mathbf{G}$ is the final generated geometric figure; Φ represents the geometric figure generation function; Ψ is base quadrilateral generation function; $q \in \mathcal{Q}$, where $\mathbf{Q}$ is the predefined set of quadrilateral types; $\mathbf{P}=(p_1, p_2, p_3, p_4)$ represents vectors of initial quadrilateral vertex coordinates; $\delta \in [0, 1]$ is point deletion parameter; $\mathbf{A}=(a_1, \dots, a_n), n \in 0$ to 6 is the set of added points; ω represents the weight factor for special points (e.g., midpoints, trisection points); ρ_c and $\rho_t \in [0, 1]$ are probabilities of generating inscribed/circumscribed circles and text labels; $\mathcal{T}$ is the set of possible text labels.

All rendered points are within the coordinate axis ranging from -10 to 10, with two decimal places retained to ensure accuracy. During the final label generation, we first use the *TransData* function from Matplotlib to convert the display coordinates to pixel coordinates. Then we recalculate the pixel coordinates to a range between -10 and 10. This is because Matplotlib always auto-adds coordinate axes and padding around the image during rendering, making coordinate conversion difficult. Let $\mathbf{G}$ be a geometric shape as in equation 1, this process can be represented as:

$$\hat{\mathbf{G}} = \text{Normalize}(\text{TransData}(\mathbf{G})) \times 20 - 10 \qquad (2)$$

where $\hat{\mathbf{G}}$ is the geometric shape with final coordinates labels. Normalization refers to dividing the x and y values of the original coordinates by the width and height of the original image, respectively. This is to unify the coordinate representation of training and testing data, as our validation and test sets are manually annotated.

The length and angle distributions of the rendered lines are shown in Figure 4. The lengths are mainly distributed between 2 and 10, which will serve as the guidance for setting the perceptual ruler in slow perception.
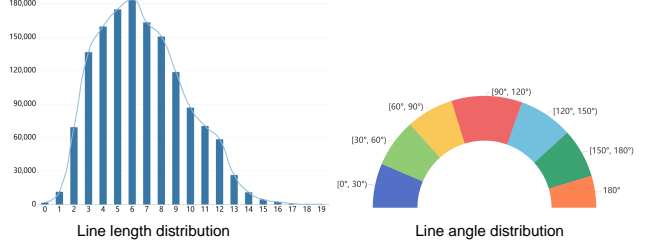

Figure 4. The line distribution of rendered train data. The left figure shows the line length and the right is angle distributions to comprise the geometric shapes in the train data.

For evaluation, we use manual manner to construct the benchmark. All images within the benchmark are sourced from the middle school mathematics exam. In total, we collect 480 geometric figures, which are divided into validation and test sets at a ratio of 1:3 via characteristics of samples, resulting in 120 images for the validation and 360 images for the test.

3.3. Slow Perception

The proposed slow perception for geometric figure parsing can be mainly divided into two stages: 1) decompose complex geometric figures into basic units and gradually perceive each one. We refer to this process as 1-order slow-down. 2) for each basic point-line pair, we use small local "perceptual jumps" to slowly and accurately reconstruct it. We name this procedure as 2-order slow-down. The detailed descriptions of them are as follows:

1-order slow down for perception decomposition. The main purpose of this stage is to unify the representation form of complex geometric shape. As shown in the input image of Figure 3, there are 8 triangles in total. If take a model to use *Tikz's* closed shape code - -*close* or Matplotlib's *Polygon* function to draw this shape, issues such as multiple peaks and redundant definitions may easily occur due to multiple triangles are nested within each other. In the 1-order slow down, we do not need to consider which figure is a polygon. We decompose all figures line-by-line, because matter how complex a geometric shape is, it always consists of basic line segments. This "make complex to simple" process can effectively avoid the multiple peaks problem via a unified representation. Using the above manner, the Figure 3 input image can be represented as:

$$\mathbf{G}' = \begin{cases} \text{Line}[\ (A, B), (B, C), (C, A), \\ \qquad (A, D), (A, E), (B, E)] \\ \text{Circle}[(C_x, C_y, R)] \end{cases} \qquad (3)$$

where $\mathbf{G}'$ is the geometric shape in Figure 3. Line and Circle are two sets that contains line units and circle units decoupled from the whole shape.

2-order slow down for perception flow. In classical computer vision fields, inference time scaling seems have always existed. For instance, processes such as the transition from proposals to the final bounding box in RPN [26], or the denoising procedure in diffusion [10], is typical coarse-to-fine inference time scaling manner. However, under the auto-regressive framework, it is not easy for LVLM to perform coarse-to-fine modeling. Therefore, we propose an alternative approach – perception flow from local to whole.

Our approach is inspired by sketching techniques. For example, when sketching portraits, there are typically two methods: One involves first constructing the framework and then gradually drawing details. This is called the Contour Method and is a coarse-to-fine approach. The other method starts by drawing details from local areas and slowly builds up to the whole. This is called the Local Method and is a local-to-whole approach.

In geometric parsing tasks, for a long straight line, humans may not draw it accurately in one stroke, and models might face similar challenges. Therefore, we define the maximum single-perception distance (perceptual ruler). Note that for such a 2-order slow-down based on "multi-stroke flow", we do not apply it to shapes other than lines in this work, due to: a) lines are the most basic and common shapes, and thus need to be prioritized; b) in geometric figures, other shapes, e.g., circle and curve, have lower interdependencies and will be our future works.

Let l be the line $\overline{AB}$, wherein the point A is the start point and the B is the end one. l can be redefined via multiple sub-lines l_i:

$$\begin{cases} l = \overline{AB} = \bigcup_{i=1}^{n} l_i, \quad \text{where} \quad l_i = [x_{i-1}, x_i]; \\ x_0 = A, \quad x_n = B, \quad |l_i| = d; \\ \forall i \in \{1, 2, ..., n-1\}, \text{ and} \quad |l_n| \leq d; \\ n = \lceil |l|/d \rceil \end{cases} \quad (4)$$

where d is a hyper-parameter representing the length of perceptual ruler. n is the number of sub line segments, composed of l and d together. Assuming $l = 12$ and $d = 8$, then $n = 2$. If $d = 4$, then $n = 3$. Therefore, when l is fixed, the smaller the perceptual ruler d, the more "strokes" are needed to trace a line, resulting in greater computational complexity and increased inference time.

3.4. Optimization and Evaluation Objectives

The input of the model is a geometry image v and the output is the parsing text sequence t. The training optimization objective is as follows:

$$\mathcal{L}(\omega, t) = -E_{(t,v) \sim D} \log P_\omega (t_m \mid t_{<m}, v) \quad (5)$$

where w denotes the target text sequence, v denotes the vision features from the vision backbone, m denotes the current index of the output target token and D denotes the
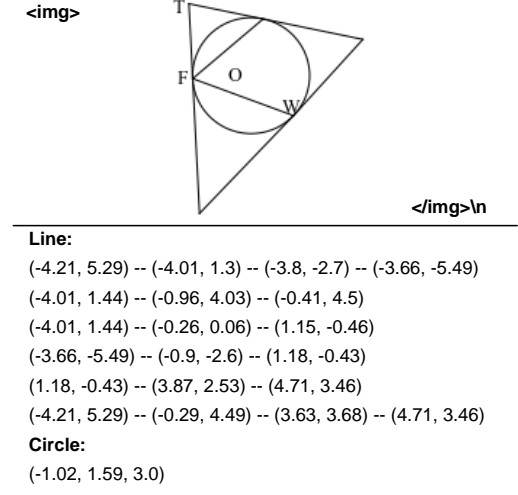

Figure 5. An example of the ground truth. This figure shows an rendered geometry sample and the corresponding text labels under the length of perceptual ruler being 4.

dataset. ω represents the model weights. An example of the input image and ground truth text is shown in Figure 5. Since we are focusing solely on the geometric figure parsing task, we do not use any prompts.

In evaluation, we use the intersection-over-union (IoU) to determine whether a predicted line segment is a positive or negative sample; specifically, the total IoU of a segment is equal to the average of the horizontal and vertical components. Mathematically,

$$\text{IoU}_{\text{line}} = \frac{1}{2} \left(\frac{|P_{\hat{x}} \cap T_{\hat{x}}|}{|P_{\hat{x}} \cup T_{\hat{x}}|} + \frac{|P_{\hat{y}} \cap T_{\hat{y}}|}{|P_{\hat{y}} \cup T_{\hat{y}}|} \right) \quad (6)$$

where P is the predicted line segment and T is the ground truth. $\hat{x}$ and $\hat{y}$ denote the the components of the line segment on the x-$axis$ and y-$axis$, respectively.

4. Experiment

4.1. Experimental Settings

Datasets: We name the train data and benchmarks we generated as SP-1, including 200k synthetic image-text pairs for trianing and 480 real-scenario samples for evaluation. We divide the evaluation part into a validation set and a test set, with a ratio of 1:3, resulting in 120 images for validation and 360 images for test. All the data we used will be open-sourced to promote the advancement of geometric figure parsing. We also hope that our data configuration will serve as the de-facto setup for subsequent followers to ensure a fair comparison.

Implementation details. We select three models for experiments: GOT [34], Qwen2-VL-2B [29], and Vary-toy [33]. GOT is the primary model for slow perception, and we conduct most of our experiments on it because it offers good

Perceptual ruler	IoU	F1	F1$_s$	F1$_l$	P	P$_s$	P$_l$	R	R$_s$	R$_l$
+∞ (baseline)	0.75	51.4	44.3	47.5	50.1	42.8	49.3	53.6	48.8	47.3
	0.9	47.5	41.6	43.7	46.3	40.1	45.2	49.5	45.9	43.6
12-length	0.75	53.3 ↑1.9	46.2 ↑1.9	49.6 ↑2.1	51.6 ↑1.5	44.9 ↑2.1	50.3 ↑1	56.0 ↑2.4	50.2 ↑1.4	50.3 ↑3
	0.9	49.9 ↑2.4	43.0 ↑1.4	47.2 ↑3.5	48.3 ↑2	41.7 ↑1.6	47.8 ↑2.6	52.4 ↑2.9	46.8 ↑0.9	47.8 ↑4.2
10-length	0.75	54.4 ↑3	48.4 ↑4.1	49.6 ↑2.1	52.9 ↑2.8	47.1 ↑4.3	50.1 ↑0.8	56.8 ↑3.2	52.5 ↑3.7	50.1 ↑2.8
	0.9	51.4 ↑3.9	45.7 ↑4.1	47.0 ↑3.3	50.0 ↑3.7	44.6 ↑4.5	47.4 ↑2.2	53.6 ↑4.1	49.5 ↑3.6	47.7 ↑4.1
8-length	0.75	55.4 ↑4	50.4 ↑6.1	49.9 ↑2.4	54.0 ↑3.9	49.0 ↑6.2	51.3 ↑2	57.7 ↑4.1	54.5 ↑5.7	49.9 ↑2.6
	0.9	52.1 ↑4.6	47.3 ↑5.7	48.0 ↑4.3	50.7 ↑4.4	45.9 ↑5.8	49.3 ↑4.1	54.3 ↑4.8	51.1 ↑5.2	48.0 ↑4.4
4-length	0.75	57.5 ↑6.1	52.4 ↑8.1	51.8 ↑4.3	55.8 ↑5.7	50.8 ↑8	52.9 ↑3.6	60.7 ↑7.1	56.9 ↑8.1	52.2 ↑4.9
	0.9	53.5 ↑6	47.3 ↑5.7	49.5 ↑5.8	51.9 ↑5.6	45.9 ↑5.8	50.4 ↑5.2	56.0 ↑6.5	51.2 ↑5.3	49.9 ↑6.3

Table 1. Results of different manners on the SP-1 **test-set**. Here, "s" and "l" are abbreviations for "short" and "long," representing short segments and long segments, respectively. The threshold is set at 8, with segments less than 8 considered as short and those greater than 8 as long. The red upward arrow ↑ indicates the improvement of the current method over the baseline at 0.75 IoU, while the blue ones ↑ signifies the performance improvement under 0.9 IoU.

Perceptual ruler	IoU	F1	F1$_s$	F1$_l$	P	P$_s$	P$_l$	R	R$_s$	R$_l$
+∞ (baseline)	0.75	52.2	41.3	49.2	51.1	39.2	50.6	53.7	46.6	48.9
	0.9	48.6	36.4	47.2	47.6	34.9	48.6	50.1	40.6	46.8
4-length	0.75	56.7 ↑4.5	44.3 ↑3	54.3 ↑5.1	54.9 ↑3.8	42.0 ↑2.8	55.5 ↑4.9	59.5 ↑5.8	49.6 ↑3	54.4 ↑5.5
	0.9	51.9 ↑3.3	39.0 ↑2.6	51.6 ↑4.4	50.3 ↑2.7	37.2 ↑2.3	52.8 ↑4.2	54.2 ↑4.1	43.1 ↑2.5	51.6 ↑4.8

Table 2. Results of different manners on the SP-1 **val-set**. The up-arrow in the figure has the same meaning as Table 1. It can be seen that the performance improvements of slow perception on the validation split are also stable.

performance, has a smaller model size, and allows for fast iteration. Qwen2-VL and Vary-toy serve as auxiliary models to provide more solid evidence for our conclusions. For GOT, we unfreeze all parameters for training. For Qwen2-VL and Vary-toy, we freeze the encoder parameters and unfreeze the LLM part for fine-tuning. All other experimental settings are identical. Specifically, we use 8 L40s GPUs for training, run 2 epochs on the SP-1 dataset, with a per-GPU batch size of 2 and a gradient accumulation of 2, resulting in a global batch size of 32. Simple data augmentations, e.g., color/lighting jitter and Gaussian noise, are utilized.

We employ cosine annealing [17] to adjust the learning rate, starting at 3e-5, with a total of 12,500 iterations and a warm-up ration of 0.003. Training GOT takes about 3 hours, Vary-toy needs 5 hours, and Qwen2-VL-2B, due to its larger encoder [22] computational cost, requires 15 hours.

Baseline definition. We do not define model-level baselines because the direct testing performance of all models on our val/test set are too low. Instead, our baseline is method-level defined. Specifically, by training SP-1, we set a perception ruler of infinite length as the baseline, meaning that for each line, a baseline model always directly regress from
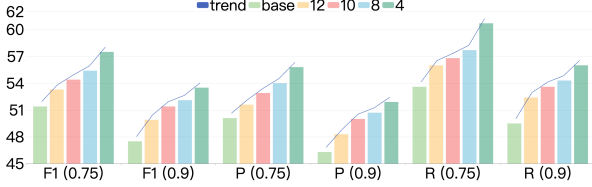
Figure 6. As the length of the perceptual ruler decreases, we can observe a steady improvement in almost all metrics. The shorter the perceptual ruler, the more "strokes" are needed to model a line, resulting in the model outputting more intermediate "gaze" points. This leads to increased computational complexity during inference, and correspondingly longer inference times, exhibiting to some extent an inference time scaling law.

the starting point to the endpoint without slow perception.

Evaluation metrics. We use F1-score, precision, and recall to measure the effectiveness of different methods. Specifically, we utilize the IoU from equation 6 to determine whether a prediction is a positive or negative sample. The basic IoU threshold is 0.75, and the strict threshold is 0.9. Precision, recall, and F1-score are defined as: $P=\frac{TP}{(TP+FP)}$, $R=\frac{TP}{(TP+FN)}$, where TP, FP and FN represent true positives, false positives, and false negatives, respectively. With the calculated precision and recall values, the F1-score can be further computed as: $F1 = 2\times(P\times R)/(P+R)$. F1-score is generally considered to be a balance between precision and recall, and it is our main metric for measuring the proposed slow perception performance.

4.2. Main Results

Effectiveness of slow perception. Table 1 shows the performance comparison of slow perception on the SP-1 test set. All results are obtained from training the GOT-OCR2.0 [34] model. The baseline (prediction line segment from the starting point to the ending point) can achieve an F1-score of 51.4%, a precision of 50.1%, and a recall of 53.6% at 0.75 IoU. When the criterion becomes stricter (0.9 IoU), these values decrease to 47.5%, 46.3%, and 49.5%, respectively. As the slow perception method is introduced, performance gradually improves. It can be observed that when using a relatively long perceptual ruler (12-length), the F1-score can be increased by 1.9%, precision by 1.5%, and recall by 2.4% at 0.75 IoU. The improvement becomes even more pronounced at 0.9 IoU. As the perceptual ruler length gradually decreases from 12 to 4, we can observe an almost steady increase in performance metrics. At a length of 4, the slow perception method outperforms the baseline by 6.1% in F1-score, 5.7% in precision, and 7.1% in recall at 0.75 IoU. These results strongly demonstrate the effectiveness of slow perception.

For images in the validation set, apart from having a different sample size compared to the test set, the geometric shapes show more complex interweaving of short lines,

meaning that short lines are more challenging to predict while long lines are slightly easier than those in the test set. As shown in Table 2, although the improvements from slow perception are lower than those in the test set, the lifts remain substantial. Slow perception mainly solves the problem of long line segments in a "single stroke" of a model through "perception flow" and results on val-set aligns with this feature, which further corroborates that the effect of the proposed slow perception is solid.

Inference time scaling law. Figure 6 presents a visual chart of the slow perception performance on the test set, which clearly demonstrates an inference time scaling law - longer inference times correlate with better model performance. This may be due to the model having an upper limit on its precise perception distance, similar to human perception. We believe that this perceptual inference scaling could also provide insights for other computer vision tasks.

Model	Size	Ruler	F1	P	R
Qwen2-VL	2B	$+\infty$	44.1	43.1	46.0
		4	46.0	45.2	47.9
Vary-toy	1.8B	$+\infty$	45.5	44.8	47.2
		4	47.8	46.7	50.0

Table 3. Slow perception on other LVLMs. We freeze the encoders to train Qwen2-VL [29] and Vary-toy [32] and test these models on SP-1 test-set to further verify the efficiency of the proposed method. The "Ruler" means the perceptual ruler length, and thus $+\infty$ represents the baseline without slow perception.

Model	Unfreeze	Ruler	F1	P	R
GOT	✓	$+\infty$	51.4	50.1	53.6
	✓	4	57.5	55.8	60.7
	✗	$+\infty$	43.8	41.7	47.3
	✗	4	46.9	44.2	50.9

Table 4. Vision encoder test. We further test whether the vision encoder is a bottleneck for geometric figure parsing task by freezing or unfreezing the GOT [34] encoder.

4.3. Ablation Study

Slow perception on other LVLMs. The above experiments are conducted based on the GOT [34] model. To verify the stability of the proposed slow perception, we select two other LVLMs for training and testing, i.e., Qwen2-VL [29] and Vary-toy [33]. Both have decoders of around 2B parameters, and we freeze their encoders during training to save GPU resources. As shown in Table 3, both models perform much lower than that of GOT in Table 1 (with the encoder unfrozen). We think the bottleneck may lie in their

original (CLIP [22]) encoders' insufficient ability to perceive geometric points and lines. Even so, slow perception still achieve a stable about 2% performance increase, which fully demonstrates its robustness.

Vision encoder bottleneck. Table 4 shows the test results after training GOT by freezing and unfreezing its vision encoder. It can be seen that unfreezing the encoder significantly improves the performance of baseline, and after unfreezing, the improvement from slow perception is even greater. This suggests that there is still considerable room in the research and training of encoders in current LVLMs.

Model	Jitter	Ruler	F1	P	R
GOT	×	4	57.5	55.8	60.7
	✓	4	56.6	54.5	59.6

Table 5. Which is more important, the accuracy of the gaze point or the perception flow? We randomly jitter the ground truth of "gaze points" along the line segment. The performance only decrease by less than 1% (57.5% vs. 56.6%).
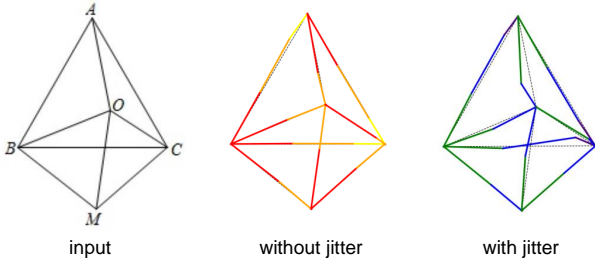

Figure 7. 'With jitter' represents the result of a trained model using gaze points that have been shaken. The "stroke order" of each line segment is mapped according to the color of the rainbow, e.g., red, orange, and yellow are used in "without jitter" result, and green, cyan, and blue are used in "with jitter" one.

Gaze points jitter. We randomly jitter the ground truth of the additional "gaze points" along line segments, with jittering ranges from 0 to 1/10 of the line segment length, to test which is more important in 2-order slow-down of slow perception: accurate prediction of gaze points or the flow of perception. From the Table 5, we can observe that adding noise to gaze points only affects performance by less than 1%. Even with imprecise gaze points, the model's performance under slow perception remains far superior to the baseline (56.6% vs. 51.4% on F1-score). This suggests that the perception flow procedure, i.e., the process of gradually perceiving from the start-point to the end-point, may be the core of slow perception. As shown in Figure 7, the accuracy of intermediate process of perceptual flow has minimal impact on the final endpoints. This conclusion mitigates the difficulty of gaze point annotation and may inspire us to extend slow perception to more general scenarios.
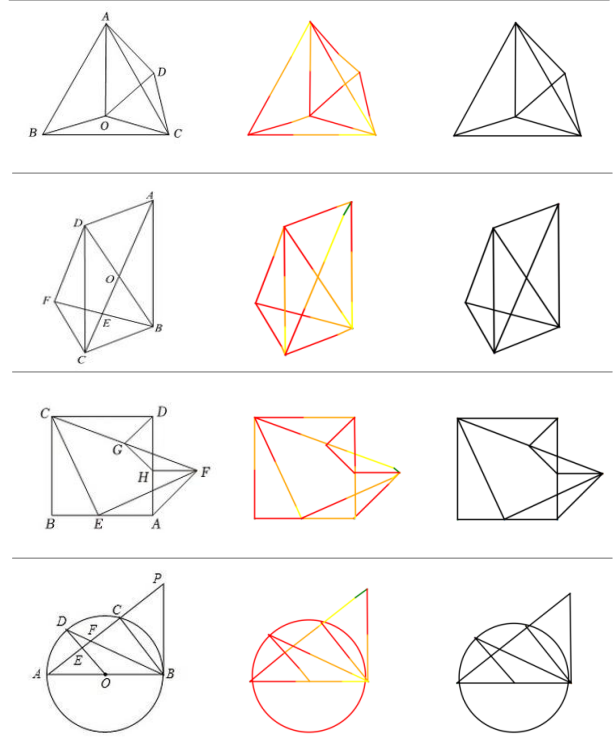

Figure 8. Slow perception visualization results. The first column represents the input image, and the second column shows the trace route of each "stroke" executed by the model in slow perception, with "stroke order" defined by rainbow colors. The third column is the final result of parsing slow perception.

4.4. Visualization Result

We provide visualization results to better understand the operation of slow perception. As shown in Figure 8, based on slow perception, the model gradually draws from the start-point to the ending when drawing each line segment by multiple "strokes". This process seems including a gradual correction process in a human-like strategy.

5. Conclusion

By consensus, "visual o1" is a promising direction and a necessary step towards AGI. However, the community seems to skip the most fundamental perception and trend to make LVLMs directly solve visual reasoning problems, e.g., mathematics in geometry. We argue that solving perception is the first step of visual o1; if a model can't even accurately copy visual geometry, how can we expect it to directly answer complex reasoning questions correctly? In this paper, we propose slow perception for geometric parsing tasks and our results show it is very effective. A geometric figure is an abstraction of natural visual scenes by humans, and we believe that our slow perception approach can also be inspiring for other general vision areas.

6. Appendix

In the main text, we primarily discuss the value of slow perception in the research field, focusing on how fine-grained perception tasks require the decomposition and flow of perception. This appendix section will further demonstrate the usage skills of slow perception in downstream application scenarios.

Because in real practical scenarios, there is a gap between the geometric images and those that we rendered for training. Therefore, we additionally add a portion of in-house real data for post-training. Note that this is only to further show our exploration of geometric parsing based on slow perception and does not affect all conclusions of slow perception in the main body.
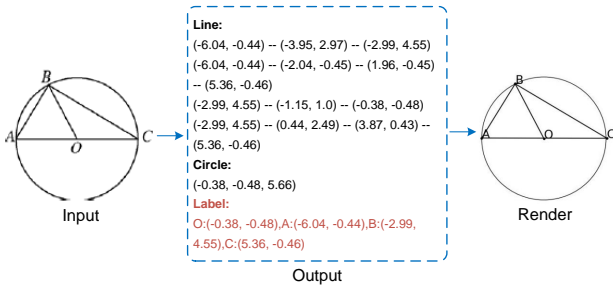
Figure 9. Adding labels for points and lines in geometric shapes is easy for the auto-regression framework. Although this process does not affect the claim of slow perception, it is necessary to embed the geometry parsing results into downstream tasks, e.g., mathematic geometric VQA.

6.1. More Complete Geometric Shapes

In geometric parsing applications, in addition to the coordinates and relationships of point-lines, sometimes we also need labels for them to support downstream business. The task itself is not related to slow perception, but since our method is based on the LVLM [34] framework, implementing this feature is very simple, i.e., you only need to simply add key-value pairs corresponding to labels in ground-truth to train the model, as shown in Figure 9.

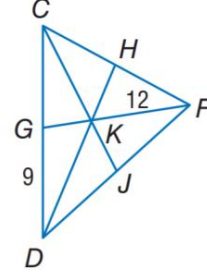### 6.2. From Geometric Parsing to Reasoning

We use Mathvista [18] Geo-subset to further verify the efficiency of geometric parsing based on slow perception for LVLMs on question-answer tasks. The Geo-subset includes 208 images. We select the state-of-the-art LVLM, GPT-4o, as the experiment target and utilize the 4-ruler slow perception GOT [34] with reality data post-training to generate a parsing reference. With the parsing results, we organize the additional reference to GPT-4o as Figure 10.

As shown in Table 6, for original results without parsing reference, GPT-4o can achieve 53.85% accuracy. When we add the parsing results as a reference, the accuracy lifts to

Model	Method	Accuracy
GPT-4o	original	53.85
	+ slow perception	60.58 ↑ 6.73

Table 6. With geometric parsing results as a reference. GPT-4o can lift 6.73% accuracy on the Mathvista geo subset. This result further indicates that even for GPT-4o, its fine-grained visual perception ability is insufficient, perception is the foundation of reasoning, and its difficulty has always been overlooked.
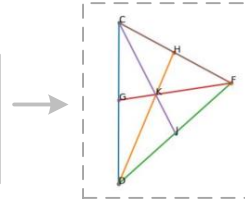
Figure 10. The organizational of input when adding geometry parsing results as a reference for GPT-4o. We provide the parsing results as a "sketch" to GPT-4o, emphasizing that it can only represent the relationship between points and lines to a certain extent, and is only for reference. We require the model that the final answer still needs to be based on the input image.

60.58%. This experimental result proves that LVLMs, even GPT-4o, suffer obvious shortcomings in perception and perception is overlooked by the community. We believe that the slow perception concept, specifically the perception inference time scaling, may be a nice solution.

References

[1] Anthropic. Claude, 2023. 2

[2] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023. 2, 3

[3] Jonas Belouadi, Simone Paolo Ponzetto, and Steffen Eger. Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. *arXiv preprint arXiv:2405.15306*, 2024. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3

[5] Jinyue Chen, Lingyu Kong, Haoran Wei, Chenglong Liu, Zheng Ge, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Onechart: Purify the chart structural extraction via one auxiliary token. *arXiv preprint arXiv:2404.09987*, 2024. 3

[6] Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric P Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. *arXiv preprint arXiv:2105.14517*, 2021. 1

[7] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024. 3

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014. 2, 3

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. 5

[11] Syrine Kalleli, Scott Trigg, Ségolène Albouy, Matthieu Husson, and Mathieu Aubry. Historical astronomical diagrams decomposition in geometric primitives. In *International Conference on Document Analysis and Recognition*, pages 108–125. Springer, 2024. 1

[12] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 1

[13] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, pages 740–755, 2014. 1, 2

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1

[15] Chenglong Liu, Haoran Wei, Jinyue Chen, Lingyu Kong, Zheng Ge, Zining Zhu, Liang Zhao, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Focus anywhere for fine-grained multi-page document understanding. *arXiv preprint arXiv:2405.14295*, 2024. 3

[16] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 2, 3

[17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[18] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023. 9

[19] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022. 1

[20] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021. 2

[21] OpenAI. Gpt-4 technical report, 2023. 2

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 6, 8

[23] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision*, pages 18–34, 2024. 3

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016. 2, 3

[25] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1, 2, 3

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 2, 3, 5

[27] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 2, 3

[28] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. 1

[29] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 3, 5, 7

[30] Haoran Wei, Xin Chen, Lingxi Xie, and Qi Tian. Cornerformer: Purifying instances for corner-based detectors. In *European Conference on Computer Vision*, pages 18–34. Springer, 2022. 1

[31] Haoran Wei, Ping Guo, Yangguang Zhu, Chenglong Liu, and Peng Wang. Humanliker: A human-like object detector to model the manual labeling process. *Advances in Neural Information Processing Systems*, 35:2294–2306, 2022. 1

[32] Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, Jinrong Yang, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Vary: Scaling up the vision vocabulary for large vision-language models. *arXiv preprint arXiv:2312.06109*, 2023. 2, 7

[33] Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, En Yu, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. Small language model meets with reinforced vision vocabulary. *arXiv preprint arXiv:2401.12503*, 2024. 2, 3, 5, 7

[34] Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, et al. General ocr theory: Towards ocr-2.0 via a unified end-to-end model. *arXiv preprint arXiv:2409.01704*, 2024. 2, 3, 5, 7, 9

[35] Haoran Wei, Chenglong Liu, Ping Guo, Yangguang Zhu, Jiamei Fu, Bing Wang, and Peng Wang. Corner affinity: A robust grouping algorithm to make corner-guided detector great again. In *IJCAI*, pages 1458–1464, 2022. 1

[36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 3

[37] Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, et al. mplug-docowl: Modularized multimodal large language model for document understanding. *arXiv preprint arXiv:2307.02499*, 2023. 3

[38] En Yu, Liang Zhao, Yana Wei, Jinrong Yang, Dongming Wu, Lingyu Kong, Haoran Wei, Tiancai Wang, Zheng Ge, Xiangyu Zhang, et al. Merlin: Empowering multimodal llms with foresight minds. *arXiv preprint arXiv:2312.00589*, 2023. 3

[39] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 1

[40] Liang Zhao, En Yu, Zheng Ge, Jinrong Yang, Haoran Wei, Hongyu Zhou, Jianjian Sun, Yuang Peng, Runpei Dong, Chunrui Han, et al. Chatspot: Bootstrapping multimodal llms via precise referring instruction tuning. *arXiv preprint arXiv:2307.09474*, 2023. 2, 3