AutoDOM: Automated Dimension Overlay For Generating Measurement Guides

Pushpendu Ghosh
gpushpen@amazon.com
Aniket Joshi
anikjosh@amazon.com
Soumyajit Chouwdhury
csoumyax@amazon.com
Promod Yenigalla
promy@amazon.com

Amazon

Abstract

Understanding product dimensions can be challenging, hindering individuals from accurately visualizing how items will fit and look within their spaces. Addressing this, we present a novel automated approach to overlay dimensional lines onto product images, empowering users to understand each subcomponent's size and scale. Our proposed multi-stage approach uses 3 key components: 3DBoundDetector to identify a bounding box around the product, QuadDetector to identify product subcomponents and AlignMatic, a post processing algorithm to determine optimal placement of dimensional lines to be overlaid. Additionally, we devise an AutoQA mechanism which ensures high-quality and accurate dimension lines by filtering aesthetically poor and incorrect dimension overlays, achieving 91.14% acceptability rate at 50% actionability, thus significantly elevating the customer experience. We benchmark our methodology against state-of-the-art image generation techniques and present ablation studies emphasizing each component's importance within our pipeline.

1 Introduction

Visualizing product dimensions accurately is a common challenge, hindering individuals from making informed decisions about how items will fit and look within their spaces, whether for personal use, design projects, or other purposes. Textual descriptions often confuse individuals. For instance, the term "arm height" can be ambiguous. Does it refer to the distance from the floor to the armrest, or the seat to the armrest? Similarly, "item height" might describe the product's upright or flat position, creating confusion for users.

Overlaying dimensions on the product image can improve the product visualization by providing clear, immediate context about size and proportions, enabling individuals to understand the scale of items relative to their environment at a glance. This visual clarity eliminates the ambiguity that arises from text-only descriptions, allowing users to better assess whether a product meets their specific needs and preferences. Existing solutions, such as

3D models and provided dimensional images, lack scalability and comprehensive coverage. Thus, a scalable automated solution for overlaying dimensional entities on images is highly desirable. To address this, we propose a novel multi-step solution comprising 3DBoundDetector, QuadDetector, and AlignMatic to overlay both outer and intricate dimensional lines directly on the main product image. Additionally, a robust Quality Assurance (QA) model automatically filters out low-quality images, guaranteeing high accuracy and user satisfaction. This approach has the potential to revolutionize the product visualization experience, empowering individuals to make informed decisions with confidence.

2 Related Works

The field of computer vision has witnessed remarkable advancements in object detection, driven by the development of deep learning models like R-CNN [6], Fast R-CNN [5], Faster R-CNN [15], and Cascade R-CNN [1]. These models excel at identifying objects within rectangular bounding boxes, achieving high accuracy and efficiency. YOLO [14] and EfficientDet [17] emerged as a popular choice due to their speed and accuracy, suitable for real-time object detection tasks. Researchers extended object detection to text detection (OCR), where models like EAST [20] and Textboxes++ [11] accurately detect text within images. Recently, large models such as Grounded DINO [12] and mVIT [13] have pushed the boundaries of object detection, enabling open-world object detection where any object can be identified based on textual prompts.

Conventional approaches often struggle with irregular shapes or orientations, thus motivating researchers to explore methods for detecting objects with arbitrary orientations, such as ReDet [7] and Oriented R-CNN [19], which integrates rotation-invariant features and regression techniques. A step further, works like Quadbox [9] and quadrilateral scene text detectors [18] detect curvy or skewed text using quadrilateral bounding boxes, predicting corners through regression techniques and specialized network architectures. Semantic segmentation provides a more granular approach by classifying each pixel into a specific category, offering a detailed understanding of object shapes within images. Popular models like DeepLab [2], Mask R-CNN [8] and SAM [10] demonstrated remarkable performance across various domains, enabling accurate boundary detection of each components within images.

We propose a novel method for generating dimensional images by automatically overlaying dimension lines and values directly onto product images. Our method employs a multi-stage pipeline: first, a 3D bounding box detector identifies the overall product dimensions; next, a quadrilateral detection algorithm segments individual components, such as the seat, backrest, and armrests of a chair or sofa; finally, a post-processing algorithm determines optimal placements for dimension lines and arrows, clearly illustrating entities like item height, seat depth, and backrest height. We also introduce an automated quality assurance (AutoQA) system to ensure accurate component identification and dimension values, resulting in a highly reliable and sophisticated solution for generating dimensional product images.

3 Problem Statement

Let $A_C = \{a_1, a_2, ..., a_n\}$ represent the set of relevant dimensional entities (e.g., seat depth, backrest height, etc. for Chair) for a specific product category C. Given a product image

I_p of product p belonging to category C, our goal is to develop a system that automatically overlays the image with line segments and arrows, visually representing each dimensional entity $a_i \in A_C$.

4 Methodology

Our proposed system for automatic dimension overlaying comprises of 4 key components. First, **3DBoundDetector** estimates a 3D bounding box that tightly encapsulates the main product. Next, a **QuadDetector** identifies 2D quadrilateral regions corresponding to each intrinsic dimensional entity. **AlignMatic**, subsequently determines the optimal positioning and formatting of dimension lines to be overlaid on the product image. Finally, an **AutoQA** mechanism estimates the confidence level of the generated image, automatically filtering low-quality results. This pipeline enables fully automated dimension overlay without any human intervention.

4.1 3DBoundDetector: For Bounding dimensional entities

The initial stage of our pipeline predicts a 3D bounding box encompassing the product, which is later used to overlay bounding dimensional entities, i.e., width, depth and height. For this task, we employ a lightweight approach utilizing EfficientNet [16], a highly efficient image classification network. The final layer of the EfficientNet model is replaced with a fully connected layer, outputting a vector of size $D = 8 \times 2$. This vector encodes the (x_i, y_i) coordinates of the 8 vertices of the cuboid in the 2D image. As our objective is single-3D object detection, we bypass the complexities of multi-object detection frameworks like YOLO-6D, opting for a simpler and more efficient approach. During inference, the edges of the predicted bounding box are slightly expanded to avoid collisions, and line segments are plotted to visualize the product dimensions.

4.2 QuadDetector: For Intricate dimensional entities

We address the task of localizing the start and end points of lines representing intricate entities by detecting the circumscribing quadrilateral for each of them. Even in cases where the object's form is curved, we define the maximum span as the entity, hence making quadrilateral a suitable geometric representation. To achieve this, we introduce a novel methodology capable of detecting arbitrary quadrilaterals, elucidated in 4.2.1. QuadDetector employs a three-part architecture. We utilize EfficientNet as the backbone network to extract multi-scale features from both the original input image and a processed image containing the product's circumscribing cuboid (output of 4.1). These features are then efficiently fused across scales using a BiFPN network [17]. Finally, the combined features are fed into a fully connected network responsible for predicting the presence of a quadrilateral object within each grid cell, the parameters of the quadrilateral bounding box, and its corresponding entity label.

4.2.1 Formulation of Detection Label

The input RGB image, with dimensions $w \times h$ pixels, is divided into a grid of $k \times k$ cells, where each cell spans $w/k \times h/k$ pixels. For each cell, indexed as i where $1 \leq i \leq k^2$, the

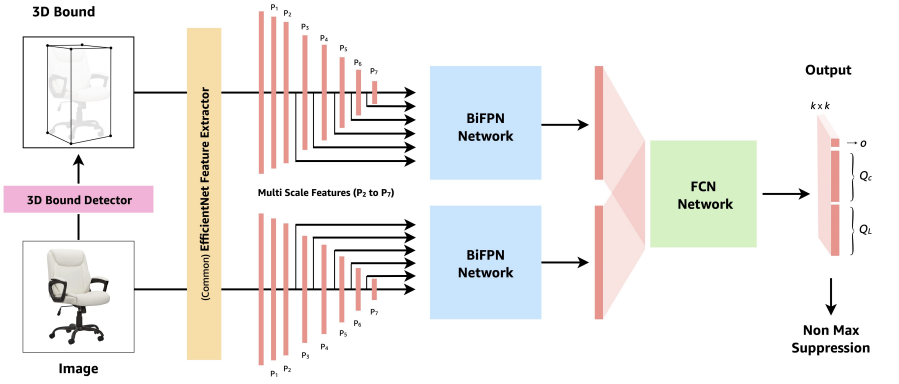


Figure 1: Model architecture of QuadDetector: Input image is passed through 3DBound-Detector to get the bounding cuboid, which after post processing is passed to QuadDetector as a second channel. The QuadDetector takes two input channel and extracts quadrilaterals corresponding to each intrinsic entity by using Non-Max suppression algorithm.

model predicts a maximum of one quadrilateral. In cases where a cell falls within multiple quadrilaterals, the nearest one (based on $\mathcal{R}^2$ distance) is assigned. The model output for the i^{th} cell, consists of three components:

1. Location vector (Q_L^i): A 11-element tuple, $Q_L^i = \{x_i, y_i, \beta, V_j \mid j \in 1,2,3,4\}$, defines the quadrilateral's location. Here, x_i and y_i represent the x and y offsets from the center of the i^{th} cell to the quadrilateral's centroid G, respectively. β is the angle made by the ground truth ideal dimensional line with the horizontal. $V_j = (d_j, \theta_j)$ represents the vector connecting the j^{th} vertex to the centroid G, with j indicating the vertex number. d_j denotes the length of vector V_j, and θ_j, normalized by 2π (ensuring a range between 0 and 1), represents the angle between vector V_j and the horizontal axis.

2. entity classification vector (Q_C^i): This vector represents a one-hot encoding of the intrinsic entity or category of the quadrilateral. The length of this vector is $1 + |A_C|$.

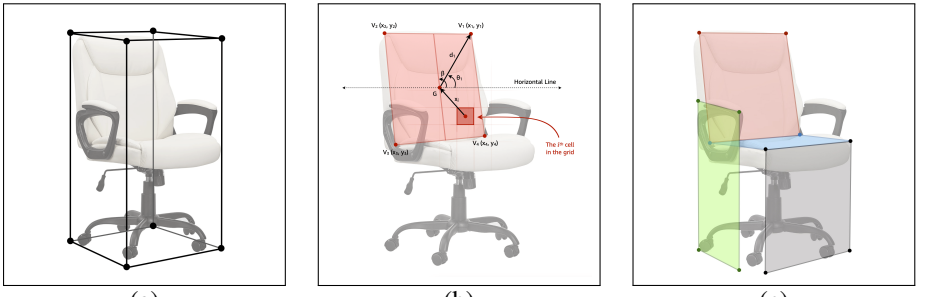
3. Objectness score (o_i): Indicating the confidence level of a object existing within a predicted quadrilateral. $(0 \leq o_i \leq 1)$

Thereby the model produces an output with dimensions $k \times k \times (|Q_C| + |Q_L| + 1)$ for each image.

4.2.2 Loss function

$$L = \lambda_L \sum_{i=0}^{k^2} 1_i^{obj} \text{SSE}(Q_L^i, \hat{Q_L^i}) + \lambda_C \sum_{i=0}^{k^2} 1_i^{obj} \text{CE}(Q_C^i, \hat{Q_C^i}) + \lambda_o \sum_{i=0}^{k^2} \text{CE}(o_i, \hat{o_i}) \quad (1)$$

The defined loss function (Equation 1) guides the model by balancing three key components. Location vector (Q_L^i) is penalised using sum of square error (SSE), whereas entity classification vector (Q_C^i) and the objectness score (o_i) is penalised using cross-entropy (CE) loss. The term 1_i^{obj} denotes if any quadrilateral appears in cell i. It is defined as 1 if any



(a) (b) (c)

Figure 2: (a) Ground truth labels for 3DBound Detector (b) Encoding quadrilateral location; (c) Ground truth labels for QuadDetector. The chair is labelled using 4 quadrilaterals for backrest height (red), armrest height (green), seat depth (blue), seat height (grey)

ground truth quadrilateral intersects with the i^{th} cell, else 0. Weights are assigned to each component to allow a balanced and optimized learning process, ensuring accurate quadrilateral detection and classification alongside robust objectness prediction.

4.3 AlignMatic: Optimal Intricate Dimensional Line Placement

Intrinsic entity lines might pass over the important product parts in the image and may look visually cluttered and not appealing. To prevent this, AlignMatic identifies the optimal placement of dimensional lines. Given a detected quadrilateral with a specific orientation, we define its left and right support lines as L and R, respectively. The family of all potential dimensional lines, D, can then be expressed as:

$$D = \{\alpha L + (1-\alpha)R \mid -M \leq \alpha \leq 1+M\} \tag{2}$$

where $M \leq 0.2$ represents a margin extending beyond the support lines, within which the dimensional line can reside. While any value of α within the defined range $(-M \leq \alpha \leq 1+M)$, accurately represents the entity, we introduce an optimization function to determine the placement that maximizes visual clarity and coherence. It involves:

1. Alignment Preference: Lines aligned with the left, center, or right support exhibit superior visual clarity. Therefore, it prioritizes α values of $\{-M, 0.5, 1+M\}$.

2. Collision Avoidance: To prevent visual clutter and ambiguity, the set of optimal lines for all entities, $\mathcal{A}^* = \{\alpha_i^* | 1 \leq i \leq |A_C|\}$, must not intersect with each other.

3. Depth Continuity: We leverage depth information to ensure dimensional lines traverse through smooth and continuous surfaces, avoiding abrupt transitions between object components or background regions. Utilizing a depth map generated by DepthAnything model, we minimize the CV (coefficient of variation) of depth values within the neighborhood pixels of the optimal line D_i for each entity a_i, denoted by $\text{CV}(\mathcal{N}(D_i))$. This constraint discourages lines from passing through multiple depth discontinuities, resulting in a cleaner and more intuitive visualization.

$$f(\mathcal{A}^*) = k_A \sum_{i=1}^{|A_C|} \mathcal{A}\text{llign}(\alpha_i) - k_C \sum_{i=1}^{|A_C|} \sum_{\substack{j=1 \\ j \neq i}}^{|A_C|} \text{Collide}(D_i, D_j) + k_D \sum_{i=1}^{|A_C|} \text{CV}(\mathcal{N}(D_i)) \tag{3}$$

Finally, we maximize f, to search for the optimal placement variable $\mathcal{A}^*$.

4.4 AutoQA

While automated dimension overlay systems offer scalability and broad product coverage, they are susceptible to visual and dimensional inaccuracies stemming from detection errors and inconsistencies in product dimension data. To address this challenge, we introduce AutoQA – an automated quality assurance framework for evaluating and filtering generated dimension images. This framework comprises two core components: Prediction Confidence and Dimension Scale Quality Filter, working together to ensure the accuracy and visual appeal of dimension overlays.

4.4.1 Prediction Confidence

We estimate the confidence of QuadDetector by computing two values. First, **Monte Carlo Dropout Uncertainity [9]:** Dropout is randomly applied to neurons within the detector's fully connected layers across multiple forward passes during inference. This process generates a distribution of predictions for each input, and the Co-variance (CoV) of this distribution, serves as a measure of model uncertainty for the detector. Second, we estimate **entity confidence** as $p_i = \max(\texttt{softmax}(Q_C^i)) \times o_i$ for QuadDetector, where Q_C^i represents the predicted class probabilities for the i-th quadrilateral and o_i denotes the predicted objectness score. Finally, these factors are combined into a quality score (QS): $QS = \frac{p_i}{k+\text{CoV}_i}$, where k is a constant. This score balances entity confidence with model uncertainty, providing a comprehensive measure of prediction quality.

4.4.2 Dimension Scale Quality filter

To address the issue of incorrect dimensional values present in the dataset (can lead to overlaying incorrect dimensional values in the images), we implement two filtering mechanisms. These mechanisms include statistical outlier detection and ratio consistency analysis.

Statistical outlierness: For each product category and entity pair, we analyze the distribution of entity values present in the dataset. Values falling outside the 5^{th} and 95^{th} percentiles are considered outliers and are excluded from further processing.

Ratio consistency: Let $\mathcal{D}_\mathcal{I}$ represent the depth map generated by the DepthAnything model, and $A_C = \{a_1, a_2, ..., a_n\}$ denote the set of all relevant dimensional entities for a given product category C. Without loss of generality, we assume a_n corresponds to the `item_height` entity, typically present across various product categories.

For each entity a_i, we compute its ratio with respect to `item_height`, i.e., $\frac{v_{a_i}}{v_{a_n}}$, forming a ratio vector $\mathcal{R}$, where v_{a_i} is the value present in the dataset for entity a_i. Under the assumption that the majority of data, after outlier removal, is accurate, we utilize this dataset to fine-tune an image classifier equipped with a regression head and mean squared error (MSE) loss. This model takes the depth map $\mathcal{D}_\mathcal{I}$ as input and predicts the expected ratio vector $\mathcal{R}$. During inference, we assess the consistency of each entity's predicted ratio $\hat{r}_i$ by comparing it to the corresponding ratio of values in the dataset. The error rate E_i for each entity a_i is calculated as:

$$E_i = \frac{1}{n} \sum_{j \neq i}^{n} \delta \left(\frac{\hat{r}_j}{\hat{r}_i}, \frac{v_{a_j}}{v_{a_i}} \right) \tag{4}$$

where δ represents a function computing the relative error between the predicted ratios ($\hat{r}_i$) and ratios of the corresponding values in the dataset. This error rate serves as a measure of inconsistency, enabling the identification and filtering of potentially inaccurate dimensional values.

5 Results and Discussion

5.1 Dataset

3D-Bounds: To train the 3DBoundDetector, we generated a synthetic dataset using ≈6.3K photorealistic 3D mesh models [3]. To ensure generality, we simulate random orientation (pose) of the product based on it's real-world data distributions and introducing diverse lighting conditions. By rendering 30 images per 3D model with random padding or cropping, we generated a final dataset of 180K images with ground-truth 3D bounding box coordinates.

Intrincate-7k: We introduce Intricate-7k, a dataset designed for detecting quadrilaterals corresponding to intricate entities of furniture products. The dataset comprises 6,916 angled-front images across 14 furniture categories. Each image contains human-annotated quadrilateral bounding boxes encompassing individual intricate entities, as visualized in Figure 2. Following annotation, a post-processing step was employed to refine and filter annotations, with subsequent re-auditing to ensure quality. The final Intricate-7k dataset, containing 6,916 images and 15,686 annotated quadrilaterals, was divided into: $\mathcal{D}_{\text{TRAIN}}$ (60%) for training, $\mathcal{D}_{\text{VAL}}$ (20%) for validation and hyperparameter tuning, and $\mathcal{D}_{\text{TEST}}$ (20%) for final evaluation.

5.2 Evaluation metrics

Quantitative metric: We calculate MAPE (Mean Average Percentage error) for 3D bounding box detection and mean average precision (mAP) between the predicted and expected quadrilaterals to measure the QuadDetector performance on the test set.

Human evaluation: The final generated images with the overlayed lines and dimension values are sent to audit to a set of 6 human auditors. Every image is sent to 3 human auditors and all auditors are asked to rate the generated image, 0 (fail) or 1 (pass), based on 2 criteria: 1. Correctness: If the overlaid dimensional lines correctly represent the intricate or bounding box dimension entity; 2. Aestheticness: If the generated image looks clean and acceptable to human. We use the majority vote per image and provide an average acceptance rate.

5.3 Results

Backbone network: We evaluated EfficientNet-B0 to B6 as backbone networks, adjusting input image size accordingly ($512 + \phi$, where ϕ corresponds to the EfficientNet version). As shown in Table 1, both mAP_{80} and mAP_{90} saturate after EfficientNet-B4. A probable reason might be that the quadrilateral labels are significantly big in size with respect to the image and thereby increasing the image size does not enhance the correctness of the model. Secondly the limited size of the dataset might not be enough to train the larger networks.

3D Bound Image as Secondary Channel: Investigating the role of the 3D Bound Image (Section 4.1) as a secondary input channel, we observed significant drop in mAP (Table 2)

Table 1: Impact of Backbone network on QuadDetector performance

Backbone	FLOPs	mAP_{80}	mAP_{90}
EftNet-B0	2.8B	68.5	55.4
EftNet-B1	6.7B	73.4	59.8
EftNet-B2	12B	75.5	61.4
EftNet-B3	27B	77.1	62.5
EftNet-B4	58B	78.8	**63.0**
EftNet-B5	140B	**78.9**	62.7
EftNet-B6	235B	78.5	**63.0**

Table 2: Impact of 3D-Bound Image as secondary channel. We present two methods (with EfficientNet-B4 as backbone): the Single Channel approach, which does not use the 3D-Bound Image, and the proposed methodology.

Approach	mAP_{80}	mAP_{90}
Single Channel	69.4	54.6
Dual Channel	**78.8**	**63.0**

and slower convergence in training upon removing it. This underscores its importance in providing crucial information on object orientation and viewpoint.

5.4 Experiments: Alternative to QuadDetector

To the best of our knowledge, no existing methods do direct dimension overlay, and hence we compare our proposed approach with several adapted methodologies:

1. **Rectangular Object Detection:** We fine-tuned YOLOv7 on a modified Intricate-7k dataset where quadrilaterals were converted to rectangles. Subsequently, a post-processing step detects the best appropriate angle of the overlay line using the output of the Euclid model and thereby draws the intricate lines inside the bounding box.

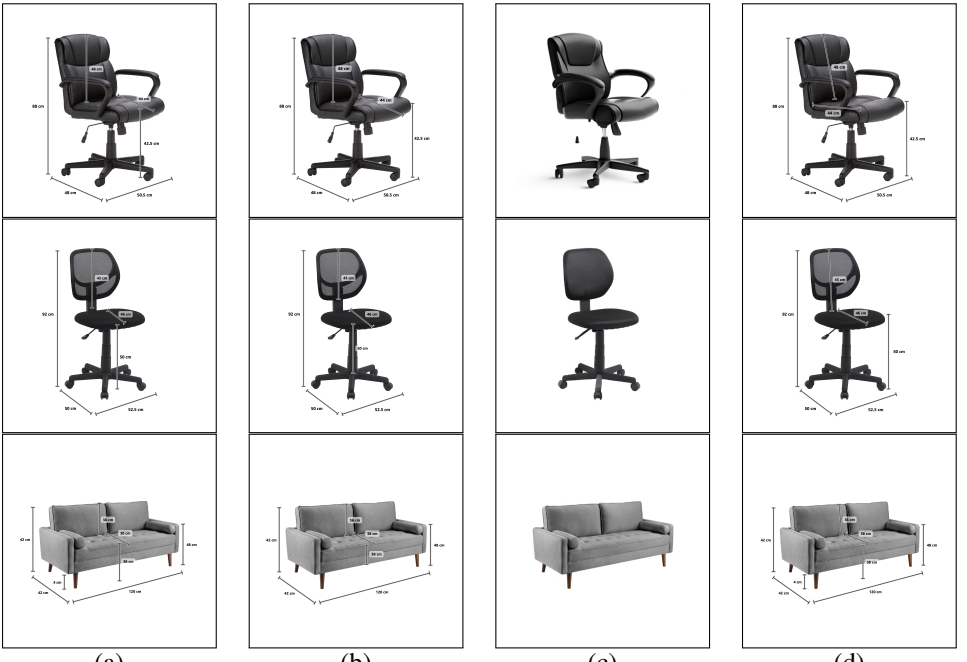
2. **Semantic Segmentation (YOLO + SAM):** We use the detected bounding boxes from YOLOv7 and apply SAM to obtain precise segmentation within those boxes. As a post processing, the endpoints are chopped or extended to cover the segmented region.

3. **Direct Image Generation with Stable Diffusion Editing:** We utilized LEDITS, a technique for editing real images via the Stable Diffusion latent space. The main image was input to LEDITS with prompts to generate lines representing specific intricate entities (e.g., backseat height, seat thickness).

We evaluated 140 generated images from each method through human evaluation (see 5.2). Example outputs are shown in Figure 3, and quantitative results are summarized in Table 3. Image acceptability denotes acceptance of all dimensional entities on the product, while entity acceptability refers to acceptance of individual dimensional entities.

Table 3: Human evaluation result on the discussed alternative methods

Approach	Image Acceptability	entity Acceptability
Rectangular Object Detection	24.51% ±4.5%	45.60% ±8.9%
Semantic Segmentation (YOLO + SAM)	30.92% ±5.8%	58.64% ±10.1%
Image Generation with SD Editing (LEDITS)	0%	0%
Proposed Methodology (w/o AutoQA thresholding)	**67.92% ±4.1%**	**86.58% ±9.2%**



(a) (b) (c) (d)

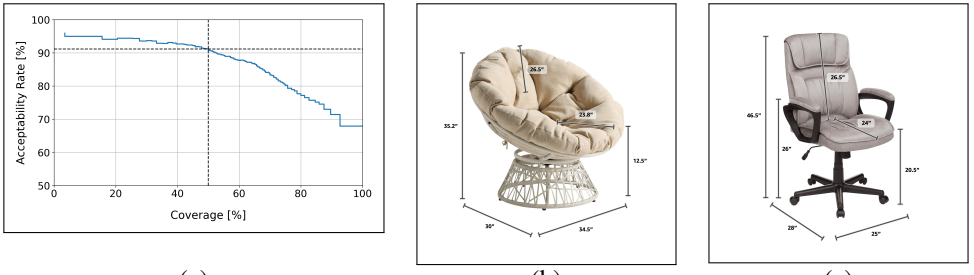
Figure 3: Comparison of dimensional image generations using various approaches: (from left to right) a. Rectangular Object Detection, b. Semantic Segmentation, c. Image Editing using Stable Diffusion, d. AutoDOM (Proposed methodology)

5.5 AutoQA Results

To improve the automated acceptability rate, our pipeline includes AutoQA which rates the quality score of generation and is able to segregate poor quality generations. This allows the pipeline to increase the acceptability rate by reducing the coverage. For the proposed methodology, we audited 1400 final output images using human evaluation methodology 5.2 and also generated the AutoQA score. As shown in Figure 4, the proposed pipeline has an acceptability rate of 67.92%, but with thresholding on AutoQA score, it achieves 91.14% acceptability at 50% coverage.

6 Conclusion

In this work, we propose a novel system comprising 3DBoundDetector, QuadDetector, and AlignMatic, which effectively identifies and positions dimensional lines to represent key product entities. Furthermore, the AutoQA component ensures high-quality outputs by filtering visually unappealing and inaccurate results. Evaluation on our Intricate-7k dataset demonstrates significant improvements in human acceptability compared to alternative methods. With the ability to achieve a 91.14% acceptability rate at 50% coverage, AutoDOM has the potential to greatly enhance the user experience by providing a clear and intuitive understanding of product dimensions.



(a) (b) (c)

Figure 4: (a) Optimizing Acceptability Rate by adjusting AutoQA Score Thresholding. Our proposed pipeline achieves a 91.14% acceptability rate at 50% coverage; (b) A poor quality generation which gets filtered out by AutoQA; (c) A good quality generation

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[3] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. *CVPR*, 2022.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

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

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[7] Jiaming Han, Jian Ding, Nan Xue, and Gui-Song Xia. Redet: A rotation-equivariant detector for aerial object detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2785–2794, 2021.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] Prateek Keserwani, Ankit Dhankhar, Rajkumar Saini, and Partha Pratim Roy. Quadbox: Quadrilateral bounding box based scene text detection using vector regression. *IEEE Access*, 9:36802–36818, 2021.

[10] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[11] Minghui Liao, Baoguang Shi, and Xiang Bai. Textboxes++: A single-shot oriented scene text detector. *IEEE Transactions on Image Processing*, 27(8):3676–3690, August 2018. ISSN 1941-0042.

[12] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[13] Muhammad Maaz, Hanoona Rasheed, Salman Khan, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Ming-Hsuan Yang. Class-agnostic object detection with multi-modal transformer. In *European conference on computer vision*, pages 512–531. Springer, 2022.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[15] 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.

[16] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[17] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[18] Siwei Wang, Yudong Liu, Zheqi He, Yongtao Wang, and Zhi Tang. A quadrilateral scene text detector with two-stage network architecture. *Pattern Recognition*, 102: 107230, 2020. ISSN 0031-3203.

[19] Xingxing Xie, Gong Cheng, Jiabao Wang, Xiwen Yao, and Junwei Han. Oriented r-cnn for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3520–3529, 2021.

[20] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.