Efficient Annotation of RGB Masks for Image Segmentation using Graham Scan

Niraj Karki
Department of Aerospace Engineering
Toronto Metropolitan University
Toronto, Ontario
Email: nkarki@torontomu.ca

***Abstract*—This paper presents a novel algorithm for automatically labeling RGB masks produced by the NVIDIA Deep Learning Dataset Synthesizer (NDDS) for image segmentation tasks. The proposed algorithm leverages Graham scan to extract polygon outlines from the RGB masks, providing a rapid means to annotate objects of interest. The algorithm achieved an annotation speed of 0.28 seconds per image for a dataset consisting of 2269 images with multiple instances. The annotated dataset trained an image segmentation model using RTMDet, showcasing promising precision and accuracy in classification and localization. The results of this study show that it is possible to rapidly annotate RGB masks with our algorithm contributing positively towards synthetic data annotation for image segmentation tasks.**

***Index Terms*—Synthetic data; NDDS; computer vision; deep learning; RTMDet; image segmentation; data annotation; PyTorch; Unreal Engine, CATIA V5**

1. Introduction

Object detection and image segmentation are two popular computer vision tasks. Object detection is the task of locating instances of objects in visual data [1]. Image segmentation is the task of identifying and segmenting classes into a region of pixels [2]. Traditionally, these tasks are carried out by training a model with real data captured and annotated manually [3]. Creating datasets for this purpose is not a trivial task. Larger benchmark datasets often have detailed processes with multiple stages throughout annotation [3]. This can be much more difficult to implement in smaller projects where resources are limited.

One alternative is to use synthetic data, a widely explored area of research in computer vision allowing users to generate and annotate large volumes of data in a short time. This is typically done using a 3D development software such as Unreal Engine. The Nvidia Deep Learning Dataset Synthesizer (NDDS) is an open source plugin for Unreal Engine that is used to generate synthetic data in several industries [3]. NDDS does not support polygon annotations required for image segmentation. Instead, the software outputs a segmented RGB mask with a unique colour for each object of interest. Converting the RGB mask into an annotation format like COCO or Darknet is not a trivial task. NDDS does not yet have the ability to extract polygon outlines and write to an annotation file.

This paper proposes a novel algorithm which makes use of the RGB mask to obtain a polygon outline of the object of interest and save it to an annotation file. Our algorithm uses Graham scan to find the smallest convex hull around the region of pixels masking the object of interest. This enables rapid annotation for image segmentation algorithms which are by nature laboriously intensive to annotate.

2. Background

2.1. NVIDIA Dataset Synthesizer (NDDS)

The NVIDIA Deep Learning Dataset Synthesizer (NDDS) is a plugin from NVIDIA for Unreal Engine 4 (UE4). The platform is open-source and is widely used within several industries. The plugin supports images, segmentation, depth, object pose, bounding box, keypoints, and custom stencils [4]. The plugin includes different blueprints for generating highly randomized images. Among other things, NDDS is capable of randomizing lighting, objects, camera position, poses, textures, as well as the camera path following [4]. These tools help create variability in the training process and enable the model to interpolate to real world instances [3].

2.2. Real-Time Models for Object Detection (RTMDet)

The RTMDet-Ins-x model was selected to evaluate the efficacy of the annotated data. RTMDet (Real-Time Models for Object Detection) are a family of models capable of performing real-time instance segmentation and rotated object detection [5]. RTMDet outperforms the YOLO series of architectures, achieving an average precision (AP) of 52.8% on the MSCOCO dataset with 300+ FPS on a NVIDIA 3090 GPU [5]. RTMDet exploits large-kernel-depth-wise convolutions as the primary building block in the neck of the model.

Figure 1 depicts the instance segmentation branch of RTMDet. The mask feature head consists of 4 convolution layers that extract mask features with 8 channels from multi-level features. The kernel head predicts a 169-dimensional vector for each instance, divided into three parts with lengths of 88, 72, and 9 respectively [5].

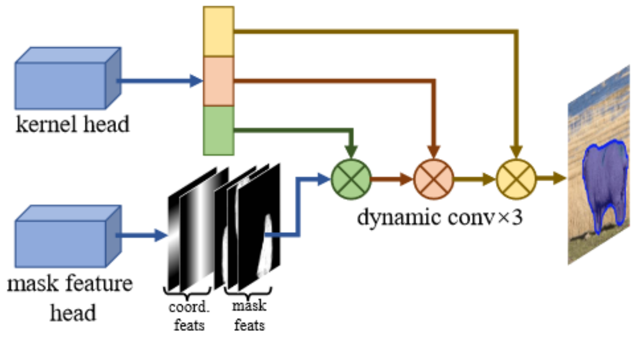

Figure 1: RTMDet image segmentation branch architecture [5].

3. Method

3.1. Apparatus

The objective of this experiment is to detect simple mechanical components from their 3D models. Using CATIA V5, two gear sizes, two shafts, and a ball bearing were modelled. These can be found in appendix A.

3.2. Data Generation

NDDS was used to generate unique instances of the model data and corresponding RGB mask. Metal textures from the automotive materials pack were tested and applied to the components to obtain photorealistic results. The camera position and pose were adjusted to obtain different views of the objects in the scene.

The image background was randomized to prevent overfitting and increase the robustness of the detection models subject to varying real-world environments. Kai et al. showed that state-of-the-art recognition models misclassify images even in the presence of correctly classified foregrounds [6]. This occurred up to 88% of the time with adversarially chosen backgrounds [6]. As a result, randomized backgrounds were added cautiously, only adjusting the colour and pattern marginally.

Param et al. used distractor objects synthetically pasted on random backgrounds. This strategy brings improvements to object detection that relies on the fine-grain appearance of an object rather than its visual context ([7], [8]). In Param's experiment, this strategy did not extend trivially and impacted the accuracy of their detector on the VOC'12 dataset [7]. So, the number of distractor objects added were limited. Figure 2 depicts the environment used to generate synthetic data. An overheard camera was used to record data of the 3D models with randomized movement and rotation in a confined control volume.

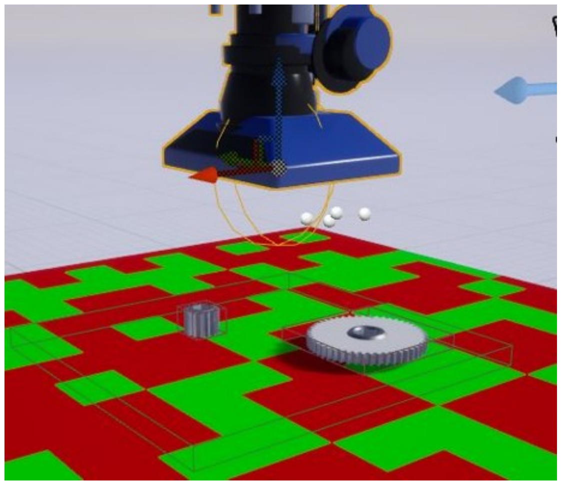

Figure 2: Virtual environment setup

Figure 3 and 4 depict sample image data and its corresponding RGB mask. Table 1 tabulates the breakdown of the dataset.

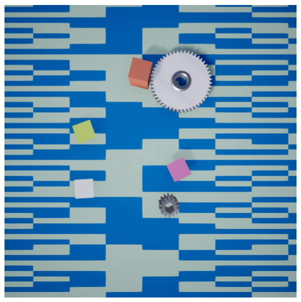

Figure 3: Image data.

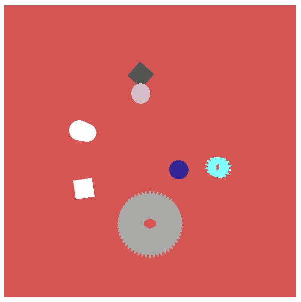

Figure 4: RGB Mask data.

TABLE 1: Image data summary for component classes.

Classes	Number of Instances	Image Size
Big Gear	1202	640x640
Small Gear	1202	640x640
Big Shaft	1067	640x640
Small Shaft	1067	640x640
Ball Bearing	1067	640x640

3.3. Data Annotation

Using the RGB mask, the following algorithm was developed to extract the polygon mask and bounding box coordinates for each object of interest. The data is dumped into a JSON file in the COCO annotation format.

1) Associate each object of interest with an RGB value, s.
2) Find the pixel coordinates (x_s, y_s) of the corresponding mask.
3) Apply the Graham scan algorithm to find the convex hull of the set of pixels in the object mask.
4) Find the bounding box of the object of interest using $[(x_{min}, y_{max}),(x_{max} - x_{min}, y_{max} - y_{min})]$.
5) Write the coordinates of the polygon mask and bounding box data to a JSON file in COCO annotation format.

The Graham scan algorithm is an efficient approach to find the convex hull of a set of points in a plane. The Graham scan returns the smallest convex polygon that encloses all the given points. The convex hull of a sample of points (i.e., pixels) is the minimum convex that encloses all points [9]. The time complexity of the graham scan is $O(nlog(n))$ [10], where n is the number of input points. Graham scan's competitive performance hinges off of its efficient sorting. The application of the Graham scan algorithm will now be discussed briefly.

Once the pixels associated are identified (as in step 1), the point, P, with the lowest y-coordinate is identified. If duplicate points are found, the one with the lowest x-coordinate is selected.

The remaining points are sorted in order of increasing polar angle from P (anchor point). This is done using equation 1.

$$\theta = \tan^{-1} \frac{x_i - x_{anchor}}{y_i - y_{anchor}} \tag{1}$$

Where (x_{anchor}, y_{anchor}) and (x_i, y_i) represents the coordinates of the anchor point and i_{th} mask point respectively. The sorting is optimized using the quicksort algorithm [1].

The convex hull is initialized with the anchor point as the first element. The sorted list is iterated, and each point is assessed to determine if traversing to it from the prior two points in the hull constitutes a clockwise (CW) or counter-clockwise (CCW) turn. For three points $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $P_3 = (x_3, y_3)$. The z-component of the cross product $\vec{P_1P_2}$ and $\vec{P_1P_3}$ is calculated from equation 2.

$$\vec{P_1P_2} \times \vec{P_1P_3} = \begin{vmatrix} i & j & k \\ (x_2 - x_1) & (y_2 - y_1) & 1 \\ (x_3 - x_1) & (y_3 - y_1) & 1 \end{vmatrix} \tag{2}$$

Where the z-component of the cross product is computed using 3.

$$k = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \tag{3}$$

If $k > 0$, the three points make a CCW turn, otherwise a CW turn if $k < 0$. If $k = 0$, then the three points are colinear.

Figure 5 depicts the results of the Garaham scan algorithm for the brown mask (outlined in red). The polygon mask does a satisfactory job of outlining the object of interest

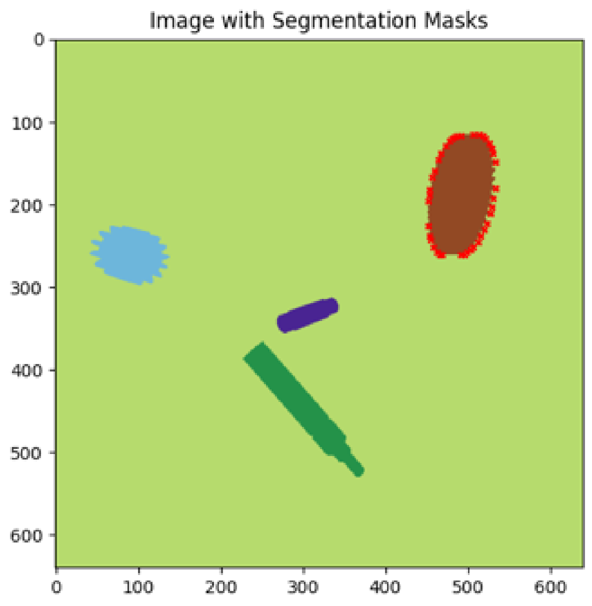

Figure 5: Graham scan example

The Graham scan imposes some limitations. Firstly, the algorithm assumes convexity. In cases where the object's shape is concave or has multiple concavities, the algorithm may not accurately delineate the boundaries leading to incorrect segmentation. Furthermore, if the object's boundary contains discontinuities or interruptions, the Graham scan might not handle these scenarios effectively. Although the time complexity of the algorithm is $O(nlog(n))$, for a large number of points the algorithm can become computationally expensive.

3.4. Testing and Evaluation

This paper uses a transfer and supervised learning approach to train the model. Table 2 tabulates the configuration parameters of the training model.

TABLE 2: Configuration parmeters of RTMDet-Ins-x model

Backbone	Deepen Factor	1.33
	Widen Factor	1.33
Neck	In Channels	[320,640,1280]
	Out Channels	320
	CSP Blocks	4
Head	In Channels	320
	Feature Channels	320
	CSP Blocks	5

3.5. Hardware Environment

Table 3 depicts the hardware setup used to train and deploy the model. For live inference, the model was deployed using an Intel Realsese L515 RGBD camera.

TABLE 3: Hardware environment.

Hardware	Specification
CPU	Intel Core 15-13600K
GPU	NVIDIA GeForce RTX 3090
RAM	32 GB DDR5 @ 6200 MHz
Operating System	Windwos 11

4. Results

4.1. Annotation Time

The average time to annotate the training dataset is depicted in 4. The algorithm yielded an average annotation time of 0.28 seconds per image.

TABLE 4: Image data summary for component classes.

Number of Images	Number of Instances Per Image	Average Time to Annotate
2269	2	0.28 seconds/image

4.2. Confusion Matrix

Figure 6 depicts the resulting confusion matrix obtained from the model's inference on real-world data. The results showcase that the model performed well enough for general component detection. The model demonstrated its poorest performance when detecting the small shaft yielding an 8% true positive rate. This result is likely attributed to a shortage of identifiable features with the small shaft.

For smaller components with more dense features such as the ball bearing or small gear, the model performed much better yielding true positives rates of 64% and 81% for the small gear and ball bearing respectively. The larger gear yielded a true positive rate of 66%. For the gear components, a better detection rate was anticipated due to the convex nature of the Graham scan algorithm. A 76% true positive rate was achieved for the larger shaft, demonstrating that the algorithm has reasonably adept annotation capability in labeling non-convex components.

It is noted that no efforts were made to generate an optimized dataset for detection purposes. There are several aspects that could be enhanced to establish a more resilient detection model. For example, Fine-tuning the model using a portion of real world data can increase the robustness of the model. Addressing domain gaps through adversarial training or advanced data synthesis techniques could further bridge the synthetic-real gap.

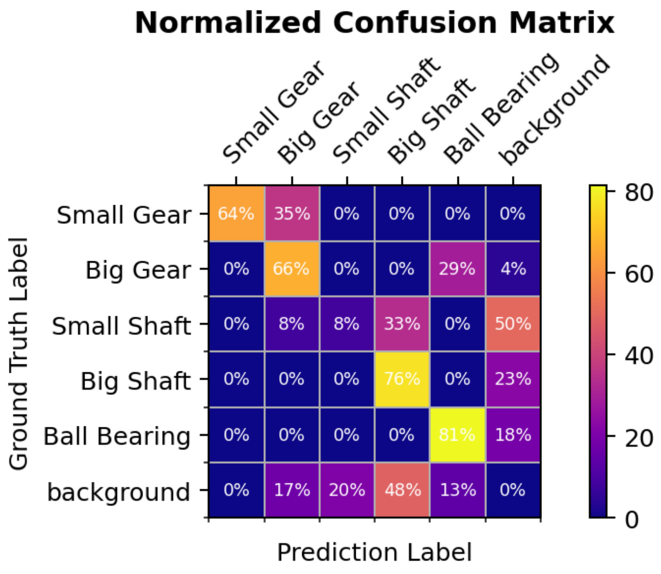

Figure 6: Confusion matrix results

Figure 7 displays the results of real-time inference in a controlled environment. The gears and small shaft are good examples of bad generalization. The model tends to mix up detections between the two gear sizes, as well as between the different shafts. Comparatively, the big shaft and ball bearing showcased more consistent and reliable detections.

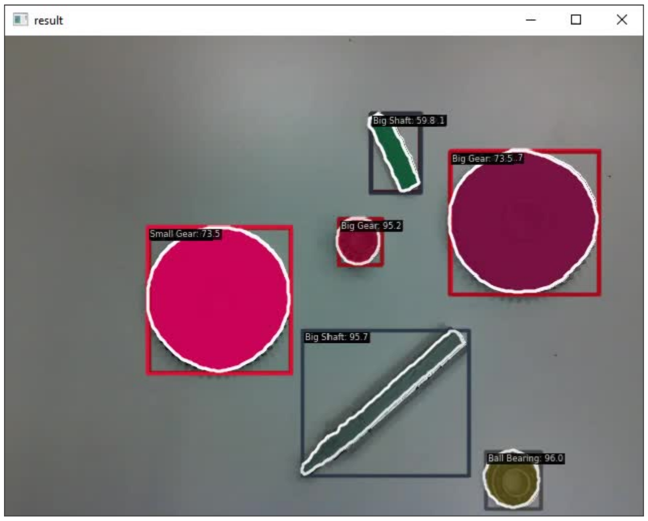

Figure 7: Live inference results. Click **here** to view video results.

5. Conclusion

The main objective of this paper was to develop a novel approach to automatically label RGB masks produced by NDDS. We wanted to examine the performance of the algorithm in terms of total time to annotate the dataset, and the accuracy of the detections. In this paper, we trained an image segmentation model using synthetic data generated on NDDS with our custom labeling algorithm. The model was trained utilizing a transfer and supervised learning approach in the CNN-based RTMDet.

The findings of this paper show that it is possible to train an image segmentation model using our customized algorithm, even with non-convex components. For a total of 2269 images, our algorithm yielded a labeling time of 0.28 seconds per image. Classes with minimal features tended to be under generalized and harder to detect in the real world. However, this problem is more likely associated with the inherent inaccuracies of synthetic data, and not the labeling algorithm itself. The training results manifest good precision and accuracy in localization and classification even with non-convex components, proving the potential for further research.

Future work will consist of optimizing the current algorithm for speed and precision. While the current algorithm yields promising results, there are numerous approaches to constructing a convex hull. The efficiency of these algorithms depend on the input number of data points and the final hull size. Therefore, the annotation efficiency can be optimized considering the size of the object of interest, leading to a potential area of research. We hope this paper can provide valuable insight for academic and professional peers exploring this field.

References

[1] T. M. Inc., “What is object detection?,” 2022.

[2] T. M. Inc., “What is image segmentation?,” 2022.

[3] I. Rasmussen, S. Kvalsvik, P.-A. Andersen, T. N. Aune, and D. Hagen, “Development of a novel object detection system based on synthetic data generated from unreal game engine,” *Applied Sciences*, vol. 12, no. 17, 2022.

[4] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, “NDDS: NVIDIA deep learning dataset synthesizer,” 2018. https://github.com/NVIDIA/Dataset_Synthesizer.

[5] C. Lyu, W. Zhang, H. Huang, Y. Zhou, Y. Wang, Y. Liu, S. Zhang, and K. Chen, “Rtmdet: An empirical study of designing real-time object detectors,” *arXiv preprint arXiv:2212.07784*, 2022.

[6] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry, “Noise or signal: The role of image backgrounds in object recognition,” *arXiv preprint arXiv:2006.09994*, 2020.

[7] N. Dvornik, J. Mairal, and C. Schmid, “Modeling visual context is key to augmenting object detection datasets,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 364–380, 2018.

[8] D. Dwibedi, I. Misra, and M. Hebert, “Cut, paste and learn: Surprisingly easy synthesis for instance detection,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1301–1310, 2017.

[9] A. Baíllo and J. E. Chacón, “Chapter 1 - statistical outline of animal home ranges: An application of set estimation,” in *Data Science: Theory and Applications* (A. S. Srinivasa Rao and C. Rao, eds.), vol. 44 of *Handbook of Statistics*, pp. 3–37, Elsevier, 2021.

[10] A. Biswas, P. Bhowmick, M. Sarkar, and B. B. Bhattacharya, “A linear-time combinatorial algorithm to find the orthogonal hull of an object on the digital plane,” *Information Sciences*, vol. 216, pp. 176–195, 2012.

Appendix A.
3D Models

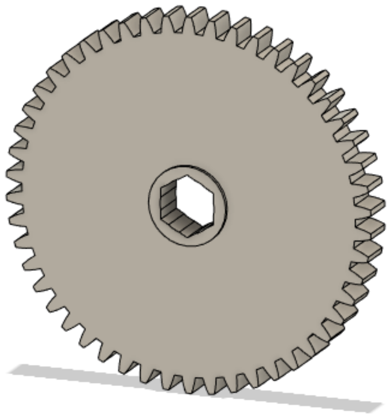
Figure 8: Big gear.

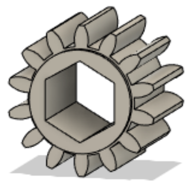
Figure 9: Small gear.

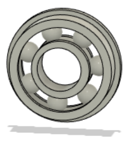
Figure 12: Ball bearing.

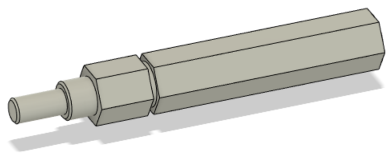
Figure 10: Big shaft.

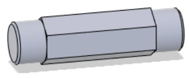
Figure 11: Small shaft.