Abstract

The paper proposes a scheme for counting the number of categories of Lego blocks based on the SAM (Segment Anything) image segmentation model. Users only need to provide Lego pictures and the number of category types expected to be classified to obtain the corresponding classification results. Our method first uses the SAM model to segment the Lego picture and extract the block mask. After that, we filter out the overlapping masks through the block mask filter based on hash similarity. Finally, we use the K-means clustering algorithm based on the area to cluster the block masks to obtain the final classification results. Our experiments were tried on the ten pictures provided. And analyzed the advantages and disadvantages of the experimental results. Thanks to the powerful segmentation ability of the SAM model, the main advantage of our method is the generality of the process, which does not depend on the distribution and geometric characteristics of the blocks in the picture, nor does it require any information other than the expected number of categories. However, the user needs to know the area sorting relationship corresponding to the expected category in advance. Under this relationship, we can map the classification results to a single certain category. The code of the experiment is attached.

本文基于 SAM(Segment Anything) 图像分割大模型提出了一种对乐高积木块的视觉图像进行类别数目统计的方案，使用者仅需要提供乐高图片和期望分类的类别种类数目，即可得到对应的分类结果。我们的方法首先使用 SAM 模型对乐高图片进行分割，提取出其中的物块掩码。之后我们通过基于哈希相似性的物块掩码滤波器过滤掉重合的掩码。最后，我们采用基于面积的 K-means 聚类算法对物块掩码进行聚类，得到最终的分类结果。我们的实验在提供的十张图片上都进行了尝试。并分析了实验结果的优缺点。得益于 SAM 模型强大的分割能力，我们的方法的主要优点在于流程的通用性，不依赖于图片中积木块的分布和几何特质，同时也不需要提供期望类别数量之外的任何信息。不过需要使用者提前知道期望类别对应的面积排序关系，在这一关系的前提下我们才能把分类结果对应到确定的单一类别上。实验的代码放到了附件中。

Contents

1 Introduction

Pre-trained on web-scale datasets, large language models (LLMs) , like ChatGPT, have revolutionized natural language processing (NLP).

And LLMs in computer vision (CV) have also achieved great success in recent days.Recently, the Segment Anything Model (SAM)[1] has achieved impressive zero-shot segmentation performance, which exhibits significant potential in open-world image perception. However, as a class-agnostic segmenter, SAM can not extract high-level semantic features, which limits its capability for openworld image understanding.In some of reacent works, some reasearchers have found that the SAM model will produced some anbiguous masks in segmenting the images. This undoubtedly proves that it is a class-agnostic segmenter. However, this does not deny the excellent potential of SAM in many image segmentation scenarios.

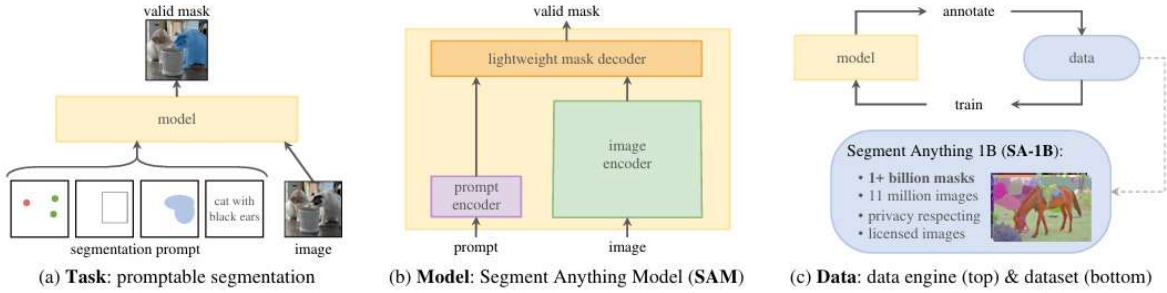


figure 1: The architecture of SAM(Figure from [1])

So one of the motivations of this paper is to explore whether it is possible to combine the excellent segmentation ability of the SAM model with the specific task requirements in the actual work scenario, and then design a set of general processes that can be used in different scenarios. So the SAM model can be used for segmentation to achieve the purpose of practical application.

figure 2: The Lego visual image

The work scenario of this paper is the segmentation and counting of Lego visual images. Lego is a very popular toy, which is characterized by the combination of different blocks to build a variety of models. In the assembly process of Lego blocks, we need to classify the blocks so that we can quickly find the required blocks during assembly. In the disassembly process of Lego blocks, we need to count the blocks so that we can quickly find the required blocks during disassembly. Therefore, the classification and counting of Lego blocks are two important steps in the assembly and disassembly

of Lego blocks. It can be expected that this automated process will play an important role in the assembly and disassembly of Lego blocks.

2 Related Work

2.1 Segment Anything Model

Recently, pre-trained with 1B masks and 11M images, Segment Anything Model (SAM) [1] emerges with impressive zero-shot class-agnostic segmentation performance. This model is created by Meta in April 5 2023.The goal of this model is to develop a prompt-based image segmentation base model, pre-trained on a wide range of datasets, to solve a series of downstream segmentation problems on new data distributions. The model structure annotated in the paper is shown in the following figure:

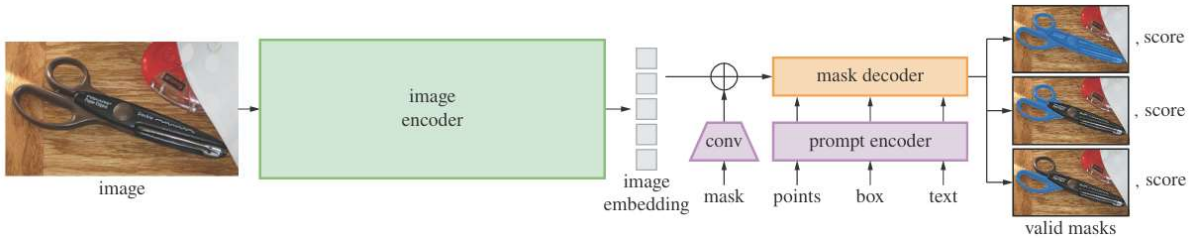


figure 3: The architecture of SAM(Figure from [1])

This model uses the ”data engine” to generate data that is beneficial to the model. Considering the problem of high artificial annotation cost in the segmentation task, the data engine of this model mainly combines the three-stage methods of manual annotation, semi-automatic annotation and fully automatic annotation to generate a large number of segmentation masks. Data, thereby improving the generalization ability of the model.

Another point is that the model uses a mask decoder inspired by NLP language large models to decode the input feature vector, which allows the model to process multiple input types of prompts such as points, boxes, and language prompts to generate segmentation masks. This provides great flexibility to this model.

The work of this paper is mainly based on the post-processing and information utilization of the SAM model to obtain image masks, so as to achieve the purpose of counting the number of blocks in the Lego visual image.

2.2 Using the Pre-trained Model for Matching

Another important work developed based on SAM is to use the pre-trained model for target matching in the image retrieval task. The core of this work is to use the pre-trained model to process the feature vector of the new input to encode the image, thereby realizing the similarity calculation of the image. However, there is a major difficulty in the task of target registration, that is, the input image in the new scene is often a small sample or even a zero sample in the data set used for training the model, which tests the generalization ability of the large model and the new scene. How do we design the task migration.

Recently, there is a work dedicated to using SAM for zero-sample migration Mather[2], which fully considers how to match the features of a single template with the features of the target image. **The filtering module in the work of this paper is inspired by this work. We use the similarity filtering module to process the obtained mask, thereby achieving the filtering of the overlapping mask.**

2.3 Automatic Counting of Lego Blocks

In the past, there have been similar works[3] based on the traditional computer vision scheme to achieve automatic classification of Lego blocks. Therefore, the work of this paper can be regarded as using a newer segmentation method based on a large model to achieve automatic classification of Lego blocks.

3 Method

3.1 Overview

Our solution is divided into three modules: **segmentation module based on SAM model, filtering module based on hash similarity and clustering module based on K-means.** By processing a picture and the expected number of categories, we can get the corresponding estimated result of the classification. In addition, if we also need to know the number of each specific category, we only need to know the area sorting relationship corresponding to the expected category, and we can map the classification result to a single certain category.

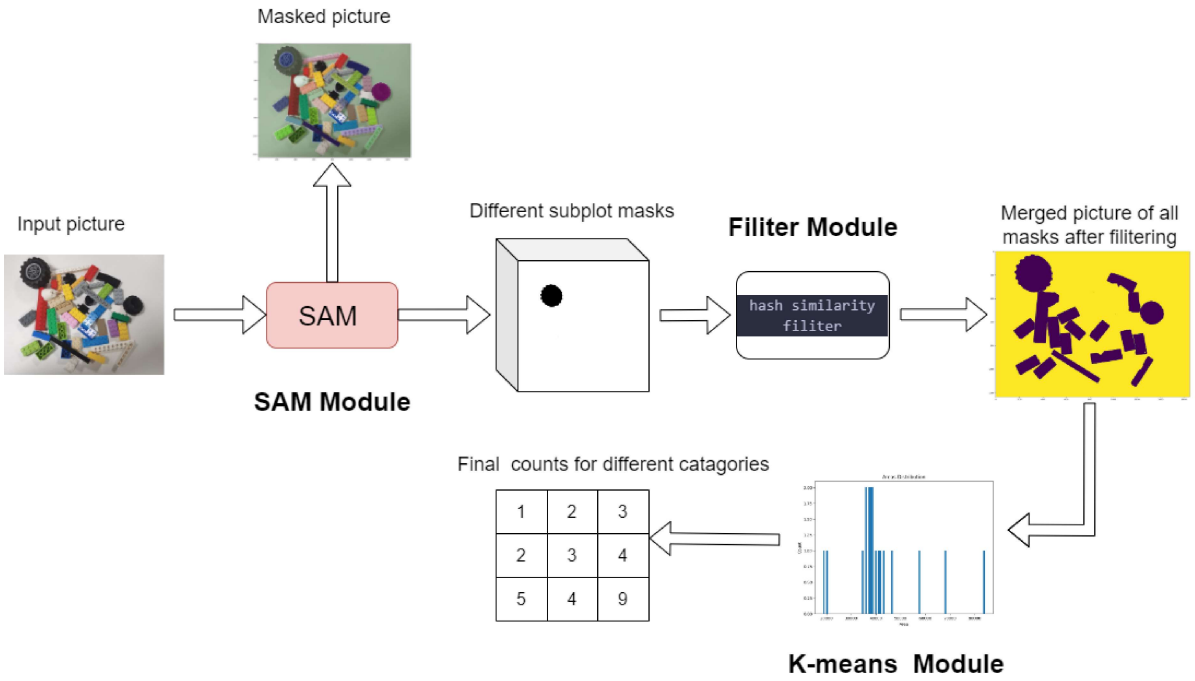


figure 4: The architecture of our method

3.2 Segmentation Module

The core of our method is based on the automatic image segmentation mask extraction of the SAM model. The SAM model uses a Vit-based image encoder to encode the input image and then import it into the pre-trained model to obtain the mask. Specifically, in the open source code, the author provides an interface for automatic mask segmentation extraction as follows:

```
mask_generator_2 = SamAutomaticMaskGenerator(
    model=sam,
    points_per_side=20,#This controls the spacing of the sampling points. The smaller the value, the
                                                    denser the sampling points
    pred_iou_thresh=0.86,#masks with iou higher than this value will be merged
    stability_score_thresh=0.98,#masks with stability score lower than this value will be filtered
                                                    out
    crop_n_layers=1,
    crop_n_points_downscale_factor=2,
    min_mask_region_area=50,  #This controls the minimum area of the mask
)
```

These parameters control the quality of the mask we expect to obtain from the SAM model. Among them, **points_per_side and stability_score_thresh are two important parameters.** They control the overlap and stability of the generated mask respectively. The smaller the sampling interval, the higher the control stability score, the higher the quality of the mask we obtain, but it will also cause the number of masks we obtain to be less. In the subsequent experiments, we adjusted the parameters for different pictures to meet the segmentation needs of different pictures.

3.3 Filtering Module

Since the masks obtained in the first stage may overlap, we need to filter these masks for subsequent clustering. Our filter is mainly based on the **perceptual hash algorithm**, which compresses and converts the image information into a hash value through **discrete cosine transform**, thereby realizing the calculation of the similarity of the image. The core idea of our filter is to convert the mask into an image, and then calculate the hash value of the image through the hash algorithm, and finally judge the similarity of the mask by comparing the similarity of the hash value. The pseudo-code of the perceptual hash algorithm is as follows:

Algorithm 1 Perceptual Hash Algorithm

Require: I: The input image
Ensure: H: The hash value of the input image
1: $I \leftarrow \text{resize}(I, 32, 32)$
2: $I \leftarrow \text{grayscale}(I)$
3: $I \leftarrow \text{dct}(I)$
4: $I \leftarrow \text{crop}(I, 8, 8)$
5: $I \leftarrow \text{quantization}(I)$
6: $H \leftarrow \text{flatten}(I)$
7: **return** H

After that, we use the obtained hash value to calculate the **Hamming distance** between any two masks, thereby realizing the calculation of the similarity of the mask. The calculation formula

of Hamming distance is as follows:

$$d_H(x, y) = \sum_{i=1}^{n} x_i \oplus y_i \tag{1}$$

Where x and y are the hash values of the two masks respectively, and n is the length of the hash value. The pseudo-code of our filter is as follows:

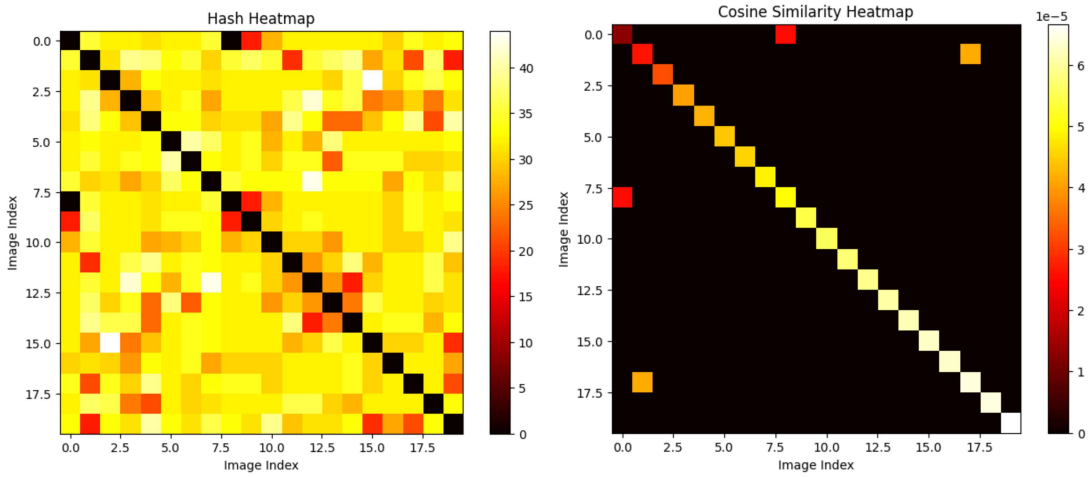
```
def pHash(img1,img2):
  img1 = img1.astype(np.uint8)*255
  img2 = img2.astype(np.uint8)*255

  #Resize the image to 32*32
  img1 = cv2.resize(img1,(32,32))
  img2 = cv2.resize(img2,(32,32))

  img1 = np.float32(img1)
  img2 = np.float32(img2)

  #DCT
  img1 = cv2.dct(img1)
  img2 = cv2.dct(img2)

  img1 = img1[0:8,0:8]
  img2 = img2[0:8,0:8]

  img1_mean = cv2.mean(img1)[0]
  img2_mean = cv2.mean(img2)[0]

  img1 = (img1 >= img1_mean)*1
  img2 = (img2 >= img2_mean)*1

  hash1 = ''.join(str(bit) for bit in img1.flatten())
  hash2 = ''.join(str(bit) for bit in img2.flatten())

  #calculate the Hamming distance
  hamming_distance = sum(ch1 != ch2 for ch1, ch2 in zip(hash1, hash2))

  return hash1,hash2,hamming_distance
```

After calculating the Hamming distance between the masks, we can judge whether two masks are similar by setting a threshold, thereby realizing the filtering of the masks. The judgment of this threshold can be adjusted by combining the hash-Hamming distance heat map of the mask segmented from the corresponding image. The left side of the figure below is the hash-Hamming distance heat map of the mask segmented from a picture.

In addition, our filter can also be designed based on the cosine similarity of the image, which is inspired by Matcher[2]. The calculation formula of cosine similarity is as follows:

$$similarity_{cos_\theta} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

Where A_i and B_i are the values of the i-th pixel of the two images respectively, and n is the total number of pixels in the image.



(a) The hash-Hamming distance heat map (b) The cosine similarity heat map

figure 5: The heat map of the mask similarity

This method can obtain more global mask similarity information, but it requires more computing resources, so we did not use this method in the overall process. Only the mask similarity heat map and hash-Hamming distance heat map calculated based on cosine similarity are shown in the figure below.

3.4 K-means Clustering Module

K-means clustering algorithm is a distance-based clustering algorithm. Its core idea is to update the clustering center continuously through iteration, so that the clustering center gradually converges to the optimal solution. The pseudo-code of the K-means algorithm is as follows:

Algorithm 2 K-means Algorithm

Require: X: The input mask k: The number of categories
Ensure: C: The clustering center
1: Initialize the clustering center C
2: **while** not converge **do**
3: $P \leftarrow$ E-step(X, C)
4: $C \leftarrow$ M-step(X, P)
5: **end while**
6: **return** C

The specific calculation formulas in the E step and M step are as follows:

$$P_{i,j} = \frac{||x_i - c_j||}{\sum_{j=1}^{k} ||x_i - c_j||} \tag{3}$$

$$c_j = \frac{\sum_{i=1}^{n} P_{i,j} x_i}{\sum_{i=1}^{n} P_{i,j}} \tag{4}$$

Where c_j is the j-th clustering center, x_i is the area of the i-th mask block, and n is the total number of mask blocks.

Considering that K-means is a relatively mature machine learning algorithm, and there are ready-made implementations in libraries such as sklearn, so we directly call the K-means algorithm

in the sklearn library here to achieve clustering of the mask. One very necessary point to explain here is: As a machine learning algorithm, the convergence of the K-means algorithm depends on the choice of the initial value and the size and distribution of the provided data, but the number of masks contained in each picture is not many, so it cannot be guaranteed. The K-means algorithm converges to the expected value accurately.

The code of the K-means algorithm we implemented is as follows:

```
def k_means_clustering(areas, k):

  #Reshape the data to fit KMeans
  data = np.array(areas).reshape(-1, 1)

  #Fit the model
  kmeans = KMeans(n_clusters=k, random_state=0, max_iter=5000, tol=1e-7).fit(data)
  #Change the algorithm's parameters to elkan
  # kmeans = KMeans(n_clusters=k, random_state=0, max_iter=5000, tol=1e-7, algorithm='elkan').fit(
                                                    data)

  labels = kmeans.labels_

  # Cluster the data points into clusters
  clusters = [[] for _ in range(k)]
  for label, area in zip(labels, areas):
      clusters[label].append(area)

  #Resort the clusters
  clusters.sort(key=lambda x: np.mean(x))

  for i, cluster in enumerate(clusters):
      print(f"Cluster {i+1}: {len(cluster)} items")

  return clusters
```

Considering that the K-means algorithm is easy to be affected by the initial value when applied to small sample data and it is difficult to converge to the expected value, we will see this in the subsequent experiments. At the same time, in order to solve this problem, we also propose two improvement plans in the subsequent discussion section. **One is to add more prior knowledge to rationalize the selection of initial values, and the other is to consider using a probability-based EM algorithm instead of the K-means algorithm.**

4 Experiments

Since our method is based on a pre-trained model, running the code requires a GPU environment. We conducted experiments on a machine with a memory of 11264MiB RTX 2090.

We conducted experiments on the ten pictures provided, including different combinations of Lego blocks, different shooting angles, different lighting conditions, different backgrounds, etc. As mentioned earlier, through the given picture and the expected number of categories, we can get the corresponding estimated result of the classification. In addition, if we also need to know the number of each specific category, we only need to know the area sorting relationship corresponding to the expected category, and we can map the classification result to a single certain category.

We used a table to record the classification results. In order to analyze the potential problems

of the method, we also saved the reconstructed pictures after filtering. It is placed in the attached folder result pictures.

4.1 The Results of All Pictures

The ten tables below show the classification results of the ten pictures respectively. The catagories here refer to the types of Lego blocks, such as 2*4 Brick, etc. For detailed category information, please refer to the appendix. **One very necessary point to explain here is: our method is an area-based recognition method, so our classification will be classified into one category with the same area, which is also a limitation of our method.**

table 1: The classification results of the first picture

Categories / Results	2*4 Brick	Sum
Estimated results	19	19
Real results	19	19

table 2: The classification results of the second picture

Categories / Results	2*2Brick	2*4Brick	2*6Brick	2*8Brick	2*10Brick	Sum
Estimated results	2	9	4	2	1	19
Real results	3	12	2	1	1	19

table 3: The classification results of the third picture

Categories / Results	1*4/2*2 Brick	1*6 Brick	2*4 Brick	1*10 Brick	2*6 Brick	2*8 Brick	2*10 Brick	Sum
Estimated results	1	17	3	2	2	2	1	28
Real results	3	2	17	1	3	1	1	28

In the classification of this picture, we found that our area may be mapped to the wrong category due to the confusion of the sample area, which is also a limitation of our method.

table 4: The classification results of the forth picture

Categories / Results	1*2 Brick	1*3 Brick	1*4/2*2 Brick	2*3 Brick	2*4 Brick	1*10 Brick	2*6 Brick	2*10 Brick	Sum
Estimated results	3	6	4	5	12	6	1	1	38
Real results	2	1	5	5	20	1	1	1	36

In this picture, we can see that our method may produce large errors when clustering mask blocks with similar areas, which is also a limitation of our method.

table 5: The classification results of the fifth picture

Categories / Results	2*4 Brick-sideways	2*4 Brick-front or back	Sum
Estimated results	2	17	19
Real results	3	16	19

In this picture, our method will classify mask blocks of the same category and different postures into different categories, which is also a limitation of our method. Therefore, our table counts two situations: 1. Mask blocks facing sideways are classified into one category; 2. Mask blocks facing the front or back are classified into one category.

table 6: The classification results of the sixth picture

Categories / Results	1*2 Brick	1*3 Brick	1*4 2*2 Brick	2*2 Brick Round	2*3 Brick	2*4 Brick	1*10 Brick	2*6 Brick	Tyre 26	2*10 Brick shape T	Tyre 18	Sum
Estimated results	2	4	3	3	4	10	5	2	2	1	1	37
Real results	1	1	5	1	2	16	1	2	1	1	1	36

From the sixth picture, we can find that simple area-based classification can no longer accurately extract most of the blocks in the provided pictures. This problem also exists in subsequent pictures, accompanied by more confusing occlusion situations.

Therefore, we no longer accurately use the mapping based on area sorting, but directly classify them into 10 categories.

It is necessary to point out that we regard the experiments of the following pictures as a proof of the generality of our model, but since the classification module K-means is essentially a machine learning method that requires a large amount of data, it is no longer possible to guarantee accurate classification on a low sample data set.

On the other hand, in order to prove that I can still segment complex blocks by using SAM, we saved all the mask subgraphs of the ninth picture in the 9masks folder.

table 7: The classification results of the seventh picture

Categories / Results	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	Sum
Estimated results	5	2	9	7	2	3	1	1	1	1	32

table 8: The classification results of the eighth picture

Categories / Results	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	Sum
Estimated results	6	5	7	3	1	1	1	24

table 9: The classification results of the ninth picture

Categories / Results	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	Sum
Estimated results	4	7	13	6	4	5	2	1	1	1	44

table 10: The classification results of the tenth picture

Categories / Results	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	Sum
Estimated results	3	6	14	7	5	4	4	1	1	1	46

4.2 Parameters

The most important parameters in the experiment consist of two parts, one is the parameters of the SAM model, and the other is the parameters of the filter.

4.2.1 Parameters of the SAM Model

The parameters of the SAM model mainly include points_per_side and stability_score_thresh, where points_per_side controls the overlap of the mask generation, and stability_score_thresh controls the stability of the mask. In order to adapt to pictures of different complexities, we adjusted these two parameters, and the corresponding parameters of the final result are as follows:

table 11: The parameters of the SAM model

Parameters \ Pictures	1	2	3	4	5	6	7	8	9	10
points_per_side	20	20	20	20	20	20	20	20	20	20
stability_score_thresh	0.98	0.98	0.97	0.97	0.97	0.975	0.97	0.97	0.96	0.96

4.2.2 Parameters of the Filter

The parameters of the filter are mainly the threshold of the Hamming distance. We adjust this threshold to control the similarity of the mask. The corresponding parameters of the final result are as follows:

table 12: The parameters of the filter

Parameters \ Pictures	1	2	3	4	5	6	7	8	9	10
Hamming distance threshold	15	15	15	15	15	12	12	12	8	8

In addition, before the mask extraction enters the filter, we set a lower limit on the size of the mask that passed through to perform preliminary screening. This lower limit setting is also a parameter. The corresponding parameters of the final result are as follows:

table 13: The parameters of the filter

Parameters \ Pictures	1	2	3	4	5	6	7	8	9	10
Mask size upper limit	15000	15000	3000	3000	15000	2000	2000	10000	8000	5000

4.3 Time Complexity

Considering that the experiment needs to be run in a GPU environment, we also record the processing time of each picture for the entire process here, as follows:

table 14: The time complexity

Time \ Pictures	1	2	3	4	5	6	7	8	9	10
Time(s)	13.7	14.0	19.0	18.6	14.8	17.1	16.5	16.0	19.8	18.1

4.4 Comments

From the trend of parameter changes, as the environment to be processed becomes more complicated, more and more Lego blocks are stacked and overlapped, and we will also set lower similarity thresholds, lower minimum area lower limits, and lower evaluation filtering indicators to ensure that our solution will not be too filtered. Overlapping mask blocks.

But the accompanying problem is that our solution will classify more mask blocks into the same category so that the clustering algorithm based on K-means cannot converge accurately to the expected number of categories.

5 More Advanced Work

5.1 The disadvantages of the current method and how could we improve

In the previous experiments, we have seen some limitations of our method, which are mainly reflected in two aspects: one is the confusion in the process of mask generation, and the other is the inaccuracy of the K-means clustering algorithm. At the same time, considering that we need to use a pre-trained model to generate masks, this will also cause our method to be affected by the pre-trained model to a certain extent and have a certain time complexity.

In addition, since our clustering algorithm is essentially a machine learning algorithm, its convergence also depends on the amount of data provided and the distribution of data, and our amount of data is limited, so we cannot guarantee the accuracy of the clustering algorithm. At the same time, our clustering index is only the area of a single mask block, which means that our method chooses to ignore the shape information of the mask of the building block, which is also a limitation of our method.

Therefore, in this part, we will try more methods with some images as experimental schemes, **especially in the segmentation scheme, clustering scheme, and the use of geometric features to propose our improvement plan.**

5.2 Pixel-based Segmentation Scheme

The advantage of introducing an image segmentation model is that it does not require better extraction of valid information from the image, but the accompanying problem is that it requires the support of hardware devices such as GPUs, and the training of the model also requires a large amount of data sets. In the process of using learning methods to process images, we generally take Patch as the basic processing unit of the image, but this often ignores the information at the pixel level. In the case of a limited number of pictures, we can consider directly taking pixels as the basic processing unit. **Inspired by the point cloud-based segmentation method in the 3D field[4], we can directly define a feature vector of a pixel, and then use a clustering algorithm to cluster the pixels, and finally map the clustering results to the mask block.** We define a **Pixel-Feature Vector** by splicing as follows:

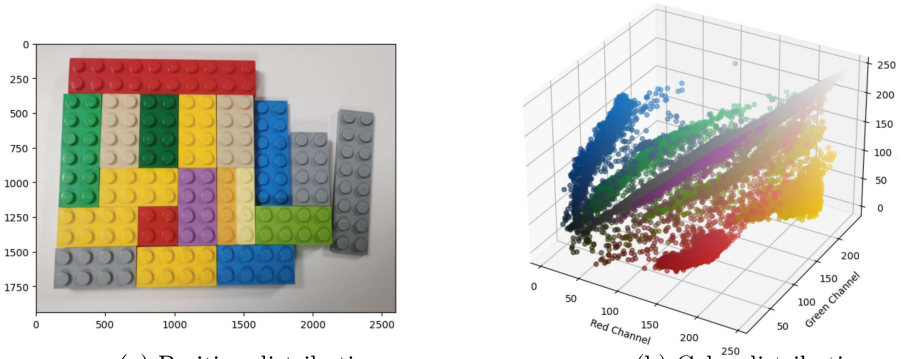
$$Pixel - Feature\_vector = [R, G, B, x, y]^T \tag{5}$$

Where R, G, B represent the normalized RGB value of a pixel, and x, y represent the normalized coordinate value of a pixel.

Once the feature vector of the pixel is defined, we can directly use the distance-based K-means clustering algorithm to cluster the pixels, and then map the clustering results to the mask block.

The following two figures show why we define the feature vector of the pixel in this way:

If we only consider the RGB distribution of the pixel, we will classify the same color and different building blocks into the same category, which is unreasonable. If we only consider the position information, we will classify the pixels of different colors in adjacent positions into one category, which is also unreasonable. Therefore, we need to consider both the color information and the position information of the pixel. At the same time, in order to balance the two considerations, we normalized the RGB value and the coordinate value.



(a) Position distribution (b) Color distribution

figure 6: The distribution of pixels

The results of clustering the second picture after processing with the above-defined feature vectors are as follows (we also compared the results of our SAM method):

table 15: The classification results of the second picture

Categories / Results	2*2Brick	2*4Brick	2*6Brick	2*8Brick	2*10Brick	Sum
SAM Segment results	2	9	4	2	1	19
Pixel-Based results	4	6	4	4	1	19
Real results	3	12	2	1	1	19

From the processing results of this picture alone, the clustering results of the Pixel-Based method still have large errors relative to the real value, and the results are far from the SAM method.**But the reason why we emphasize the key points of this idea is: once we have segmented the picture at the pixel level, we can use a limited set of pictures to establish a machine learning model that can be used for large-scale data classification or a model like the Self-Attention method, so we can use learning-based methods to processing limited-scale image data.**

5.3 Clustering Scheme Based on EM Estimation

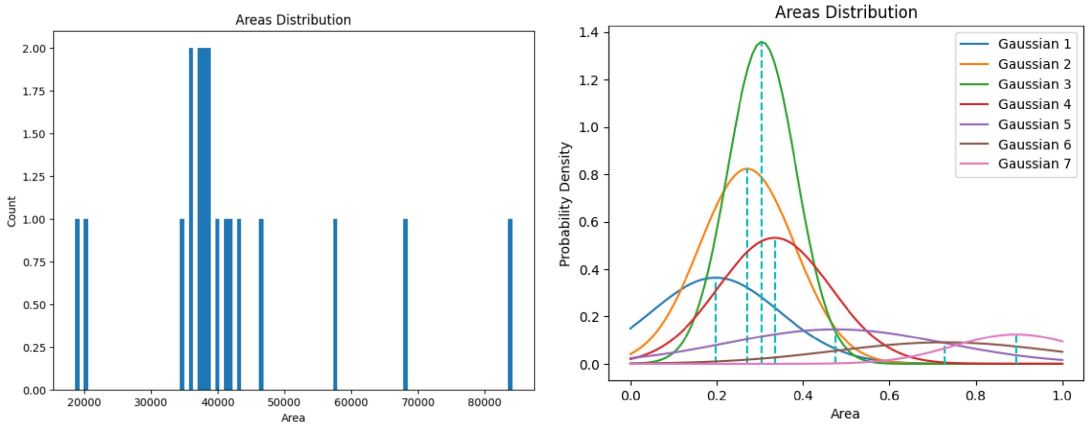
In the scheme we use, we use the K-means clustering algorithm, which is a distance-based clustering algorithm, which lacks modeling of data distribution and consideration of area error distribution. Therefore, we can consider using a probability-based clustering algorithm, such as the EM algorithm. The core of EM is to estimate the parameters of the Gaussian distribution representing each category based on the given sample, that is, we can estimate the category of each mask block in the form of probability by iterative update. The pseudo-code is as follows:

Algorithm 3 EM Algorithm

Require: X: The input mask k: The number of categories
Ensure: μ: The mean of the Gaussian distribution σ: The standard deviation of the Gaussian distribution

```
Initialize the parameters of the Gaussian distribution μ and σ
while not converge do
    P ← E-step(X, μ, σ)
    μ, σ ← M-step(X, P)
end while
return μ, σ
```

By reasonably uniformizing the initial Gaussian distribution parameters, we can converge to the expected number of categories after 2 to 3 iterations, and also ensure the area error distribution of the mask blocks of each category.



(a) The area distribution histogram (b) The theoretical k-class Gaussian distribution

figure 7: The area distribution histogram and the theoretical k-class Gaussian distribution

We conducted experiments on the second picture, and the experimental results are as follows:

table 16: The classification results of the second picture

Categories / Results	2*2Brick	2*4Brick	2*6Brick	2*8Brick	2*10Brick	Sum
K-means Cluster results	2	9	4	2	1	19
EM Cluster results	2	13	2	1	1	19
Real results	3	12	2	1	1	19

It can be seen that on this picture, the clustering results of the EM algorithm are closer to

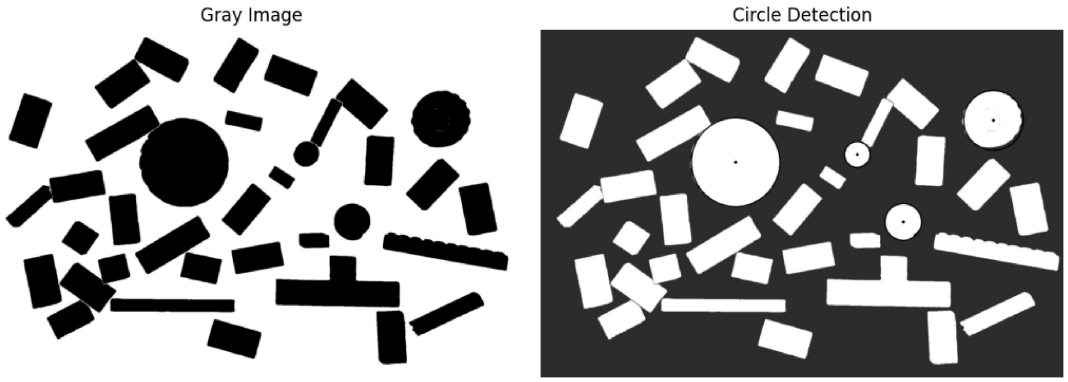
the real number of categories than the clustering results of the K-means algorithm, but for the subsequent pictures, due to the similar area distribution of different building blocks, especially a large number of building blocks with a small number of categories, Therefore, our EM algorithm cannot guarantee accurate clustering results. This is also the reason why our method does not use the EM algorithm.

In fact, it can be proved that the convergence of the K-means algorithm can be derived from the convergence of the EM algorithm, so the estimation based on EM also provides us with an idea, that is, when our sample data is sufficient or the distinction between samples is sufficient,we can consider using the EM-based clustering algorithm. At the same time, the estimation based on EM also has an advantage that it considers the distribution of **area errors**, which is impossible for the K-means algorithm based entirely on sample expansion.

5.4 Fully Utilize Geometric Features

In our experiment, since we only used the area of the mask as the only indicator for clustering, we ignored the important geometric feature information of the mask shape, which caused our clustering to fail to converge accurately to the expected number of categories when the shape became complicated. Therefore, a natural question is: Can we use the mask block that retains the most important geometric features of the building block obtained by SAM to further extract geometric feature information, and then use these geometric feature information for clustering.

Based on this, we used the **Hough transform** to extract the geometric features of the circle in the sixth picture, and the experimental results are as follows:



(a) The original picture (b) The extracted circle geometric feature

figure 8: The extracted circle geometric feature

We use the Hough transform to extract the circle in the reconstructed picture of the mask, and **mark the center of the circle** in the original picture. As can be seen from the picture in the upper right corner, our extraction result is basically correct.

6 Conclusion

In this work, we use the powerful segmentation ability based on SAM to establish a set of schemes for classifying and identifying different building blocks in Lego blocks. The core of the model is divided into three parts:**mask generation based on SAM, used to extract the mask of building blocks from the picture; filtering algorithm based on perceptual hash, used to filter out the mask that does not meet the conditions; clustering algorithm based on K-means, used for clustering the mask.** Our model's experimental results on ten pictures show that our model has a certain validity.

The biggest advantage of our solution is that our solution does not require complex preprocessing of the picture, and our solution does not require complex feature extraction of the picture, which makes our solution somewhat universal. The user only needs to know the number of categories of the building blocks to be obtained, and then adjust the parameters to get the corresponding results.

At the same time, as a supplement to the main process, we have also expanded our solution in three aspects: **one is the pixel-based segmentation scheme, the other is the clustering scheme based on EM estimation, and the third is the full use of geometric features.** We believe that if we expand in these three aspects, our solution can be applied in more scenarios. And it can also be optimized in GPU resource consumption.

Overall, although there are certain limitations in the accuracy of classification, our solution can achieve classification and recognition of Lego blocks to a certain extent, which reveals the huge application potential of the SAM model on the one hand, and also provides a certain reference for subsequent work on the other hand.

7 References

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[2] Y. Liu, M. Zhu, H. Li, H. Chen, X. Wang, and C. Shen, "Matcher: Segment anything with one shot using all-purpose feature matching," *arXiv preprint arXiv:2305.13310*, 2023.

[3] K. Wang, N. Bao, Z. Wu, and X. Ran, "Automatic separating, storing and counting system of lego building block based on machine vision," in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2017, pp. 544–549.

[4] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-nerf: Point-based neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5438–5448.