Automatic Trimap Generation for Image Matting

資工科碩 **IOC**
312551080 紀軒宇

資工科碩 **IOC**
312551089 彭筱竹

多媒體所 **ILE**
312553037 王芷鈴

1. Goal/problem

Our project aims to streamline the image composition process by developing an efficient tool. This tool simplifies the user experience by automating the generation of trimaps, a crucial component in image matting.

Traditional matting methods require manual annotation, which is time-consuming. In our approach, users input foreground and background images, and our tool automatically creates a corresponding trimap. This eliminates the need for users to prepare a trimap themselves. The generated trimap is then utilized in the matting process, followed by seamless composition with the background image. Our tool enhances convenience, enabling users to effortlessly achieve visually appealing composite images.

2. Method

Our method begins by inputting a foreground image, leveraging the DeepLab v3+ network to acquire an initial mask. Following this, we apply mask serval iterations erosion or dilation to refine or extend the mask boundaries. Subsequently, grayscale values are estimated, attributing shades to individual pixels. Utilizing this information, we generate a trimap. Employing the original image and trimap, we employ a learning-based matting method, specifically the Deep Image Matting technique, to generate the alpha matte. Finally, we achieve seamless composition of the foreground and background images using the obtained alpha matte.

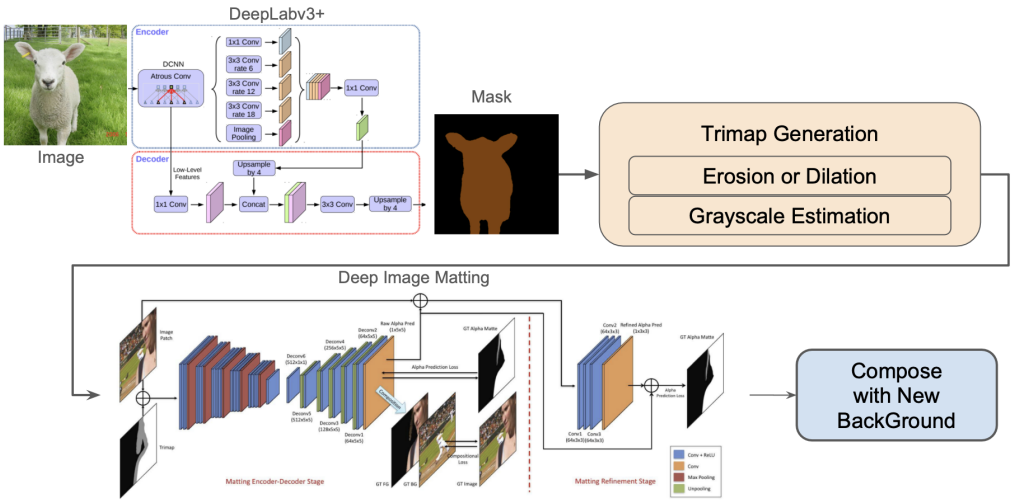

Figure 1.Structure of ours method

3. Relating Work

3.1 Segmentation

We use DeepLabv3+, which combines the advantages of spatial pyramid pooling and encoder-decoder structures for semantic image segmentation. Spatial pyramid pooling can encode multi-scale contextual information by probing the incoming features with filters or pooling operations at multiple rates and multiple effective fields-of-view. Encode-decoder can capture sharper object boundaries by gradually recovering the spatial information. Figure 2 below is the overall architecture of the model:

(1) Encoder Stage

The image will be feature extracted through a deep convolutional neural network using atrous convolution to obtain feature maps of two different depth layers. Deeper layers will use dilated convolutions with different dilation rates in ASSP for feature extraction, which improves the receptive field of the network.

(2) Dncoder Stage

The decoder part will input the two feature maps we obtained. This green feature map with higher semantic information is upsampled, and then concatenated with the shallower feature using

1*1 convolution, which is equivalent to feature fusion. Finally, it undergoes a 3*3 convolution and is adjusted to the prediction result of the same size as the input image, then obtain the segmentation result.

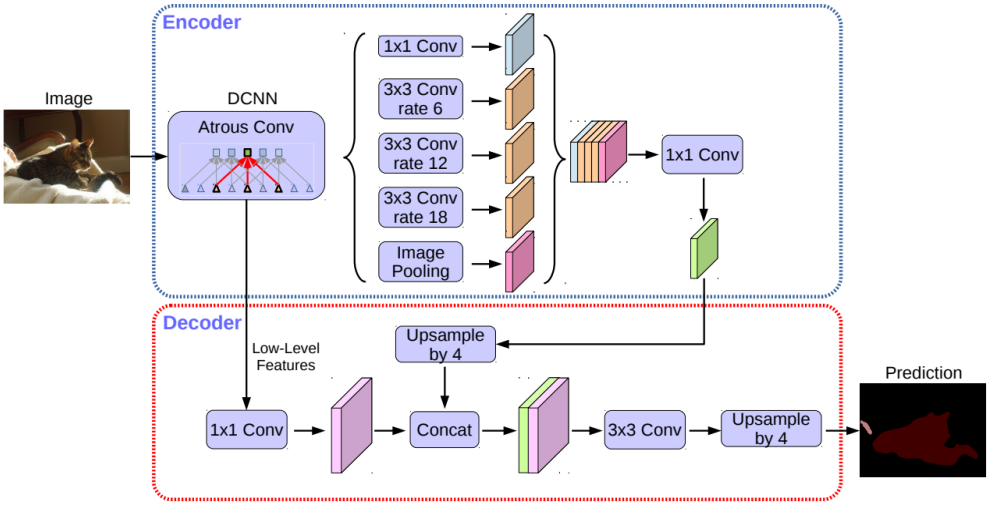

Figure 2. DeepLabv3+

There are some different models that can be used in DCNN part, such as MobileNet, ResNet50 and ResNet101, which allows us to do some experiments to compare which model has better segmentation results. Finally, we decided to use ResNet101 as DCNN for segmentation model.

3.1.1 Atrous Separable Convolution

Atrous convolution has a parameter called the dilation rate that controls the spacing between the values in the kernel. It allows the network to operate at multiple scales and capture a larger receptive field without increasing the number of parameters or the amount of computation. Atrous separable convolution apply atrous convolution in the depthwise convolution where each input channel is convolved with its own set of filters, captures spatial information within each channel, as shown in the Figure 3 below. Then use pointwise convolution after that to create a linear combination of the output of the depthwise convolution. This step learns combinations of features across different channels, allowing learning of complex representations.

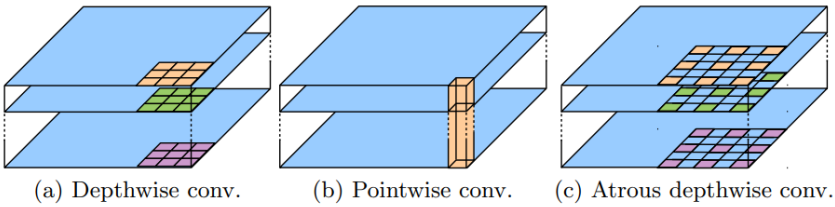

Figure 3. Atrous Separable Convolution

3.1.2 Atrous Spatial Pyramid Pooling(ASSP)

ASPP uses atrous convolution at multiple rates to capture multi-scale information without changing the resolution, as shown in Figure 4 below. We add an ASPP layer on top of the last convolutional layer. The ASPP layer pools the features and generates fixed-length outputs, which are then fed into the fully-connected layers (or other classifiers). In other words, we perform some information aggregation at a deeper stage of the network hierarchy (between convolutional layers and fully-connected layers) to avoid the need for cropping or warping at the beginning.

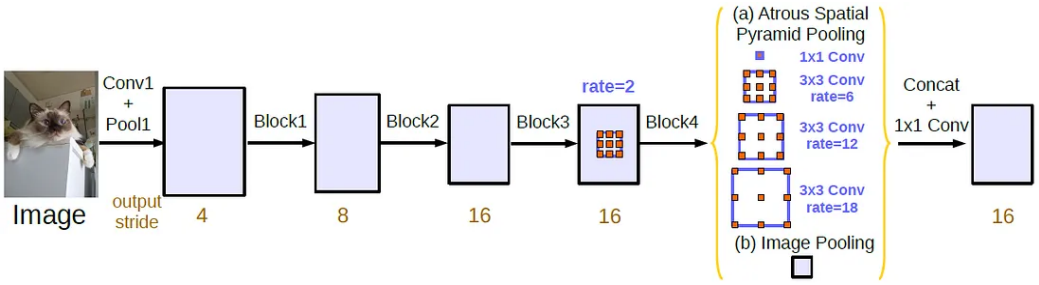

Figure 4. Atrous Spatial Pyramid Pooling

3.2 Image Matting

Image matting is a problem of estimating the foreground of an image or video, and it can be written as the equation below, where I, F, B stands for observed image, foreground image, and background image, respectively, α is the alpha matte, which is a scalar in the range of $[0, 1]$, means the opacity of the foreground, we can easily composite the foreground to another background when we have the alpha matte.

$$I = \alpha F + (1 - \alpha)B$$

3.2.1 Deep Image Matting

The previous matting method is limited since they usually solve this problem by solving the matting equation we mentioned in the last paragraph, which is a linear combination of two colors, so consqencely see this as a color problem and thus highly rely on color as the distinguishing feature, making them sensitive to the situation where the foreground and the background's color distribution overlap, for example, Figure 5 below shows the image matting using Close-Form [3], we can observe that at the right top of the image, where the hair of the troll and the bridge behind it have similar colors, the Close-Form method can not distinguish them, resulting in an imperfect matte.

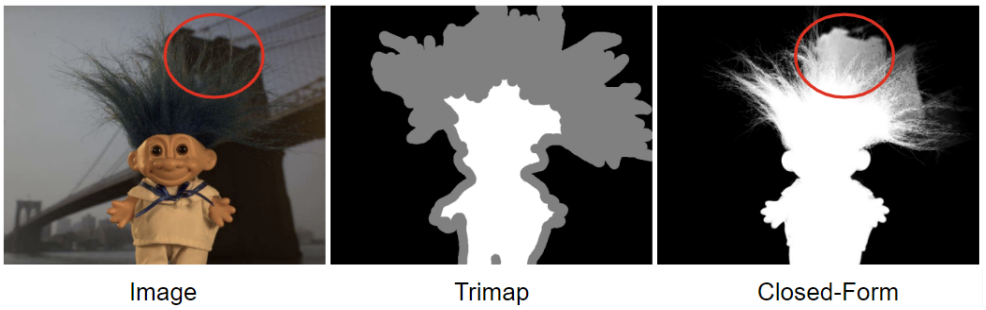

Figure 5. Image Matting using Close-Form

In Deep Image Matting, the author proposes a deep learning method to directly predict the alpha matte, using the feature extracted by the convolution neural network, which no longer needs to rely on low-level features such as color and can learn the features that possess strong structure information and texture pattern such as the hair. The network structure is shown below. The whole network consists of two stages: the encoder-decoder stage and the refinement stage.

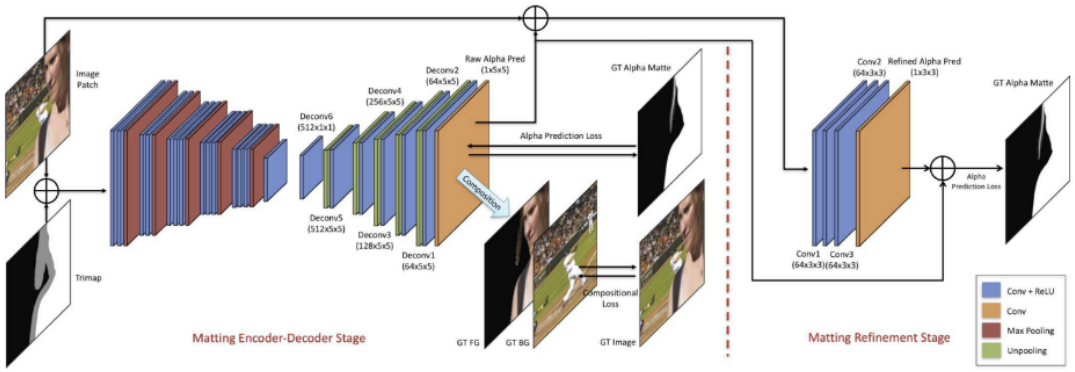

Figure 6. Structure of Deep Image Matting

(1) Encoder-Decoder Stage

The first stage is the Encoder-Decoder network, which contains several convolutional layers and max pooling layers for downsampling and several unpooling layers and convolutional layers to reverse max pooling operation for upsampling. The network takes the image patch and its corresponding trimap as input, composes them into a 4-channel image and feeds them into the network and outputs a rough alpha matte.

(2) Refinement Stage

The second stage is a fully convolutional network, which takes the image patch and the alpha matte predicted from stage 1 as input and compose them into a 4-channel image and feeds them into the network and outputs the refined alpha matte.

4. Difficulty and uniqueness

4.1 Difficulty

The primary challenge we encountered in our project lies in the intricate process of integrating multiple models. Specifically, the complexity arises from the need to harmonize different models for segmentation, trimap generation, and deep image matting.

We also found that segment model results are key, so we explored various backbones of segmentation models, including MobileNet, ResNet50, and ResNet101. After rigorous experiments, ResNet101 emerged as the most suitable backbone to help optimize our overall process. In the future, it can also be replaced with a segmentation model with better results to improve the results.

4.2 Uniqueness

Our uniqueness lies in the design of an interactive interface for our image composition pipeline. Through this interface, users can seamlessly navigate and complete the image composition step by step while preserving and storing the results at each stage.

One distinctive feature is the post-segmentation selection of foreground objects for trimap generation. Users have the flexibility to choose specific foreground elements from the segmented image,like below figure foreground objects dog and cat. Users can choose individual foreground objects or select all available options.

Figure 7. Foreground object Segmentation

Moreover, during trimap creation, users can fine-tune parameters, including grayscale estimation and dilation/erosion iterations, providing adaptable control. This uniqueness ensures that our tool aligns closely with user preferences and adjusts trimap generation based on the segmentation output, enhancing the overall customization and effectiveness of the composition process.

5. Results and comparison.

5.1 UI

We implement the whole process with a UI interface, contains the functions we mentioned in our goal, including multiple parts, shown in Figure 8, from left to right, top to down, stands for **Foreground Select/Crop, Segment, Trimap Select/Generate, Predict Alpha Matte, Foreground Select, Compose** respectively.

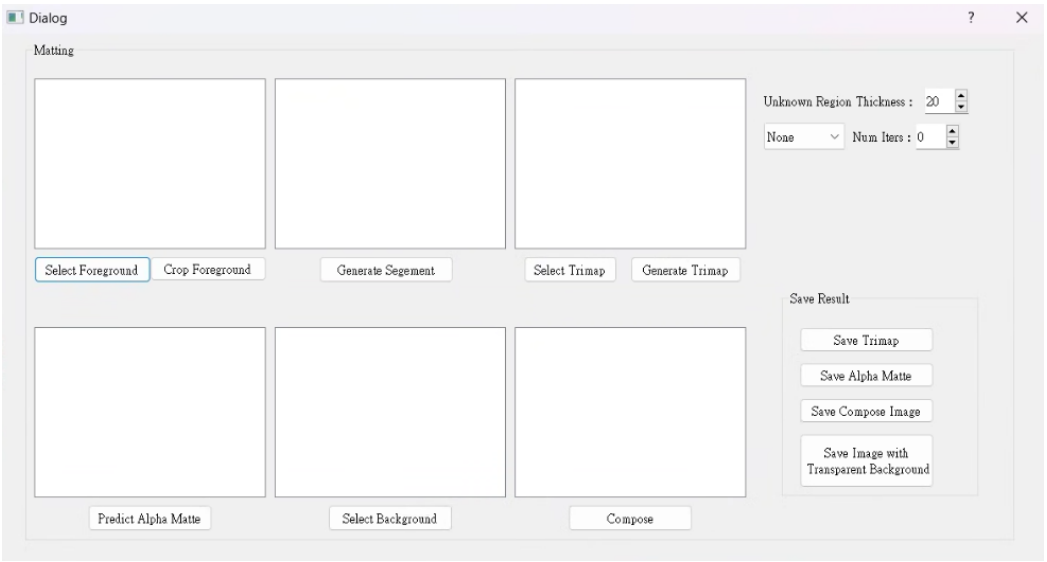

Figure 8. UI interface of our implement

5.1.1 Foreground Select/Crop

For the foreground, user can push the “Select Foreground” button to select the image for the foreground, since the segment implemented in the next step is semantic segment, in order to segment out the specific object, we also add a crop function, which will pop up a window and let user to select the region of the foreground user wants to segment in the next step.

5.1.2 Segment and Trimap Select/Generate

If user have the trimap of the foreground image already, user can simply push the “Select Trimap” button, which will do the function like the “Select Foreground” mentioned in previous part.

If not, user can generate the segment result first by pushing the “Generate Segment” button, which will do the segment by our segment model, then push the “Generate Trimap”, which will pop up a window for user to choose which class object to select as foreground object, and will generate the trimap according to the segment result and the parameter next to Trimap widget.

In this step, user can save the trimap by pushing the “Save Trimap” button.

5.1.3 Predict Alpha Matte

After having the trimap, user can push the “Predict Alpha Matte”, which will generate the alpha matte by our matting model.

In this step, user can save the alpha matteby pushing the “Save Alpha Matte” button, or save the matting result with the foreground image with “Save Image With Transparent Background”, which will simply save the result with the transparent background and save it into png format.

5.1.4 Foreground Select & Compose

After having all elements for matting, user can push the “Select Background” for choosing the foreground image and “Compose” for composing foreground and background.

In this step, user can save the composing result by pushing the “Save Compose Image” button.

5.2 Experiment

5.2.1 Exp1

Try different iterations on dilation and erosion. The figures below show the dilation and ersion with three, six, and ten iterations .We can see that when the number of dilation iterations increases, the helicopter tail becomes clearer, but the helicopter spiral exceeds the boundary.Then,when the number of iterations increases, the helicopter spiral gradually disappears. So,in different images, we can get better trimap by adjusting parameters.

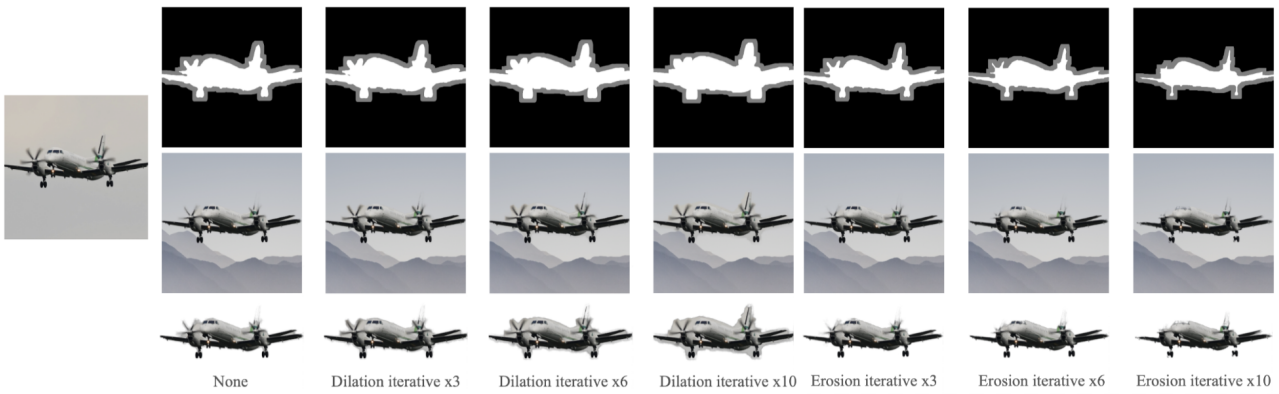

Figure 9. Results if Exp1

5.2.1 Exp2

Our experimentation involved a comparative analysis of matting results obtained using two different trimaps: one generated by our framework and another provided by alphamatting.com.

In the initial results, we observed that adjustments were necessary for optimal outcomes. We implemented an adaptive strategy, incorporating erosion adjustments and fine-tuning the gray estimation parameters. This approach resulted in improved matting outcomes. While the trimap from alphamatting.com exhibited more detailed information, our framework compensated for this by enlarging the grayscale area and leveraging deep image matting calculations.

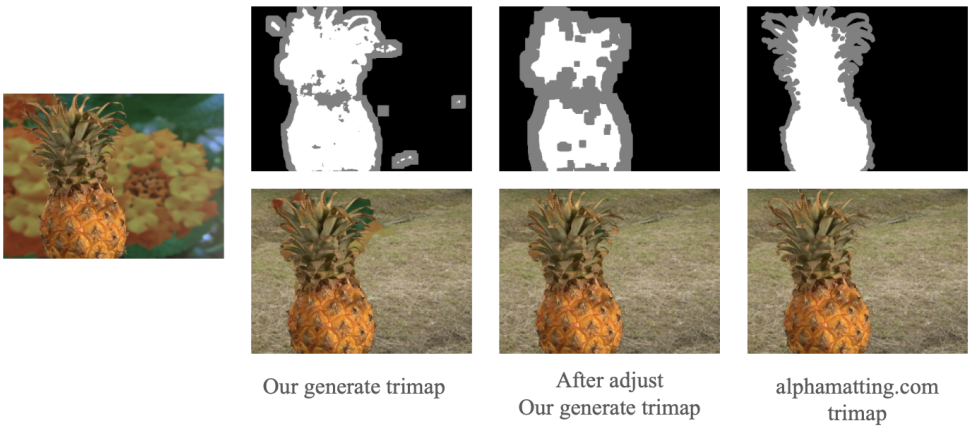

Figure 10. Results of Exp2

6. Libraries and open sources used in our project.

For the implement, we implement it with Python 3.8 and the following libraries: PyTorch, OpenCV, PyQt5, NumPy. We also use some open-source projects for some parts, including DeepLabV3+ [4], Trimap Generatiom [5], Deep Image Matting [6], and for deep learning based module, we directly use the pretrain weights of the network in the corresponding repository.

7. Contribution of each member

- 312551080 紀軒宇
  Deep Image Matting(100%)、UI design(80%)、UI function linking(80%)
- 312551089 彭筱竹
  DeepLabv3+(100%)
- 312553037 王芷鈴
  Generate trimap(100%)、UI design(20%)、UI function linking(20%)

8. Reference

[1] Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV) (pp. 801-818).
[2] Xu, N., Price, B., Cohen, S., & Huang, T. (2017). Deep Image Matting. ArXiv. /abs/1703.03872
[3] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2):228–242, 2008.
[4] *DeepLabV3Plus-Pytorch.* GitHub. https://github.com/VainF/DeepLabV3Plus-Pytorch
[5] *Generate Trimap*. GitHub. https://github.com/lnugraha/trimap_generator
[6] Liu, L. (n.d.). *Pytorch-deep-image-matting*. GitHub. https://github.com/huochaitiantang/pytorch-deep-image-matting