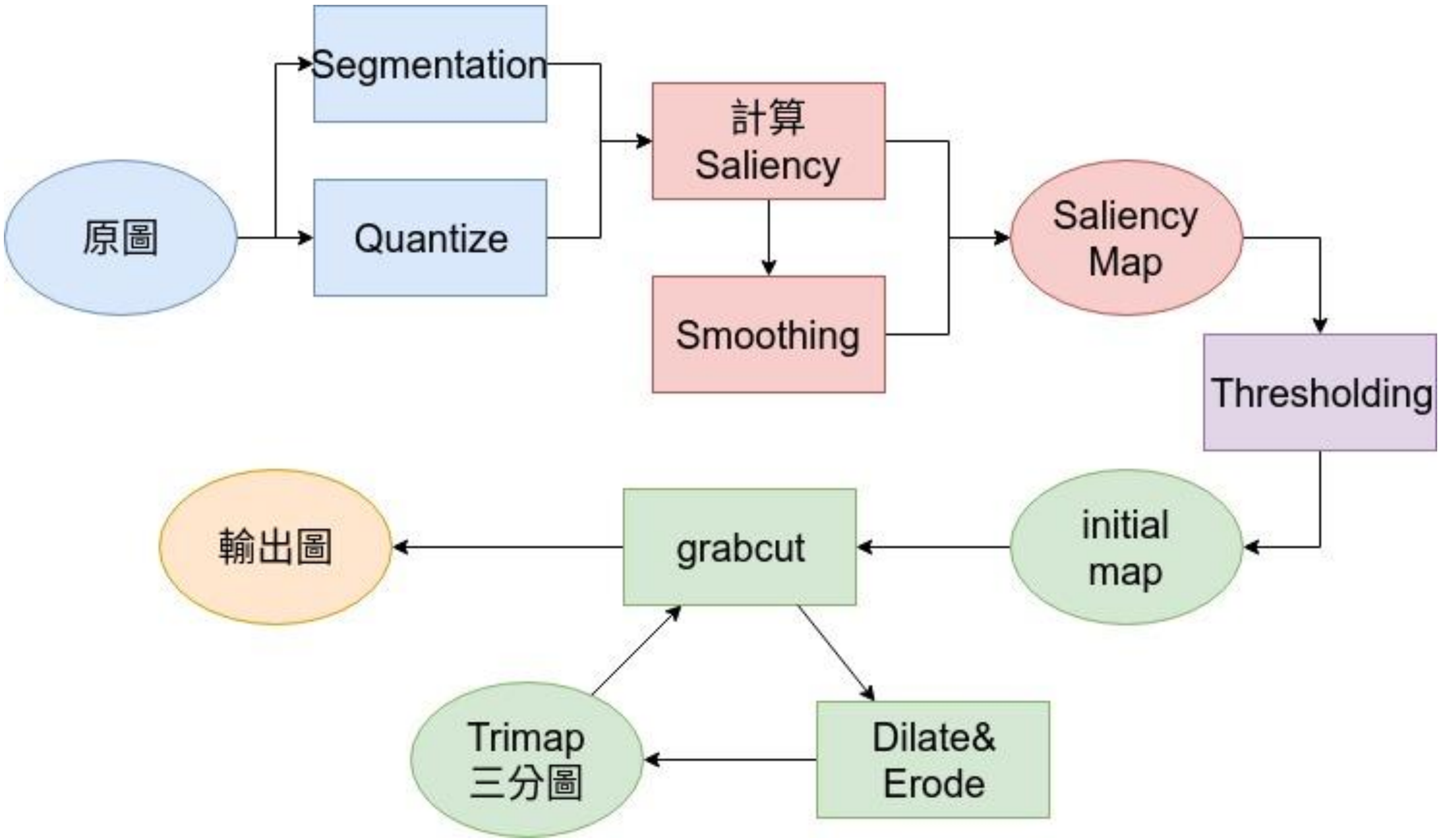
Global Contrast Based Salient Region Detection

組別 : 4

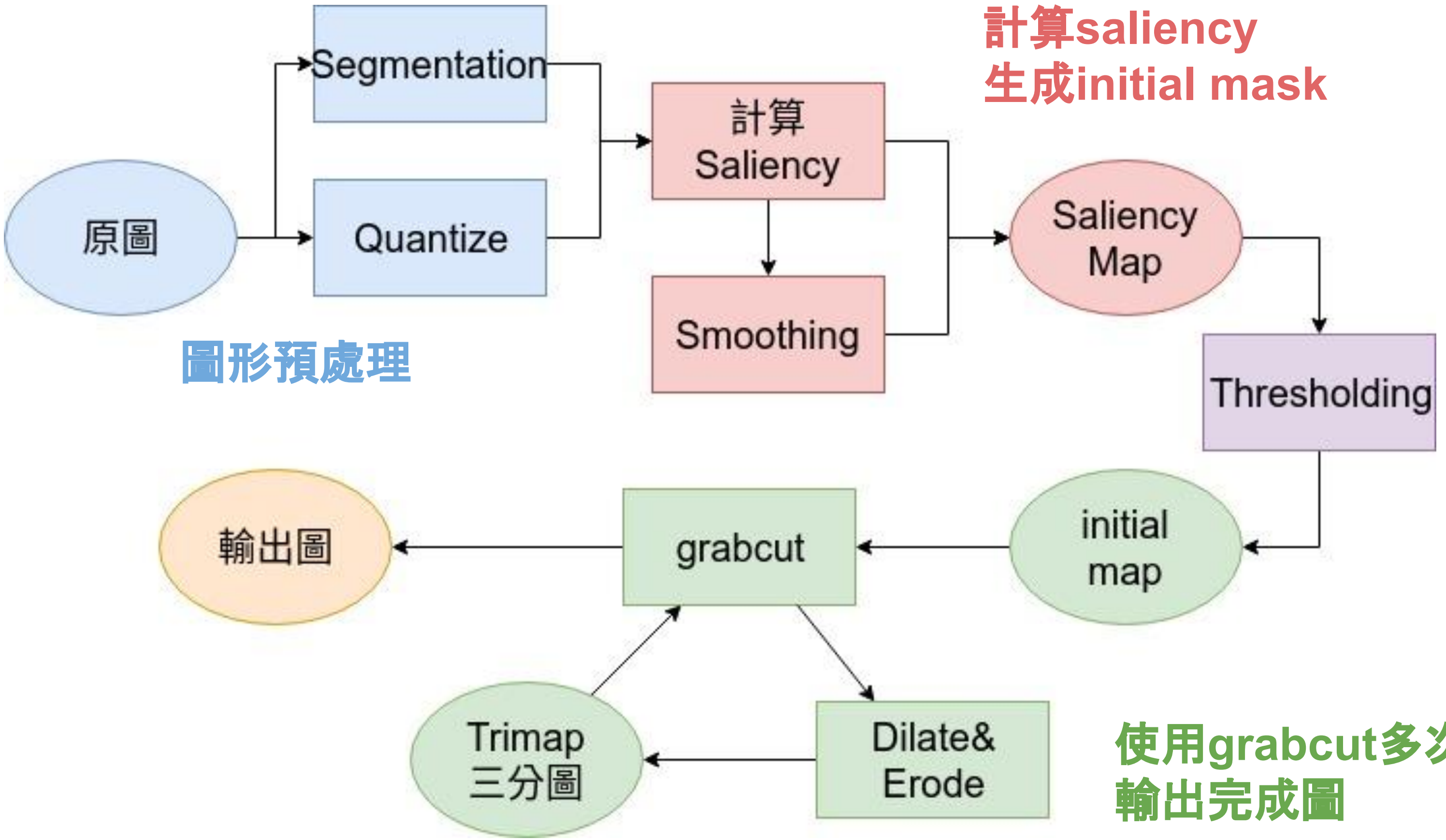
Problem definition

給定一張圖形，設法標記出該圖中的foreground object（最明顯的物體、主角）。



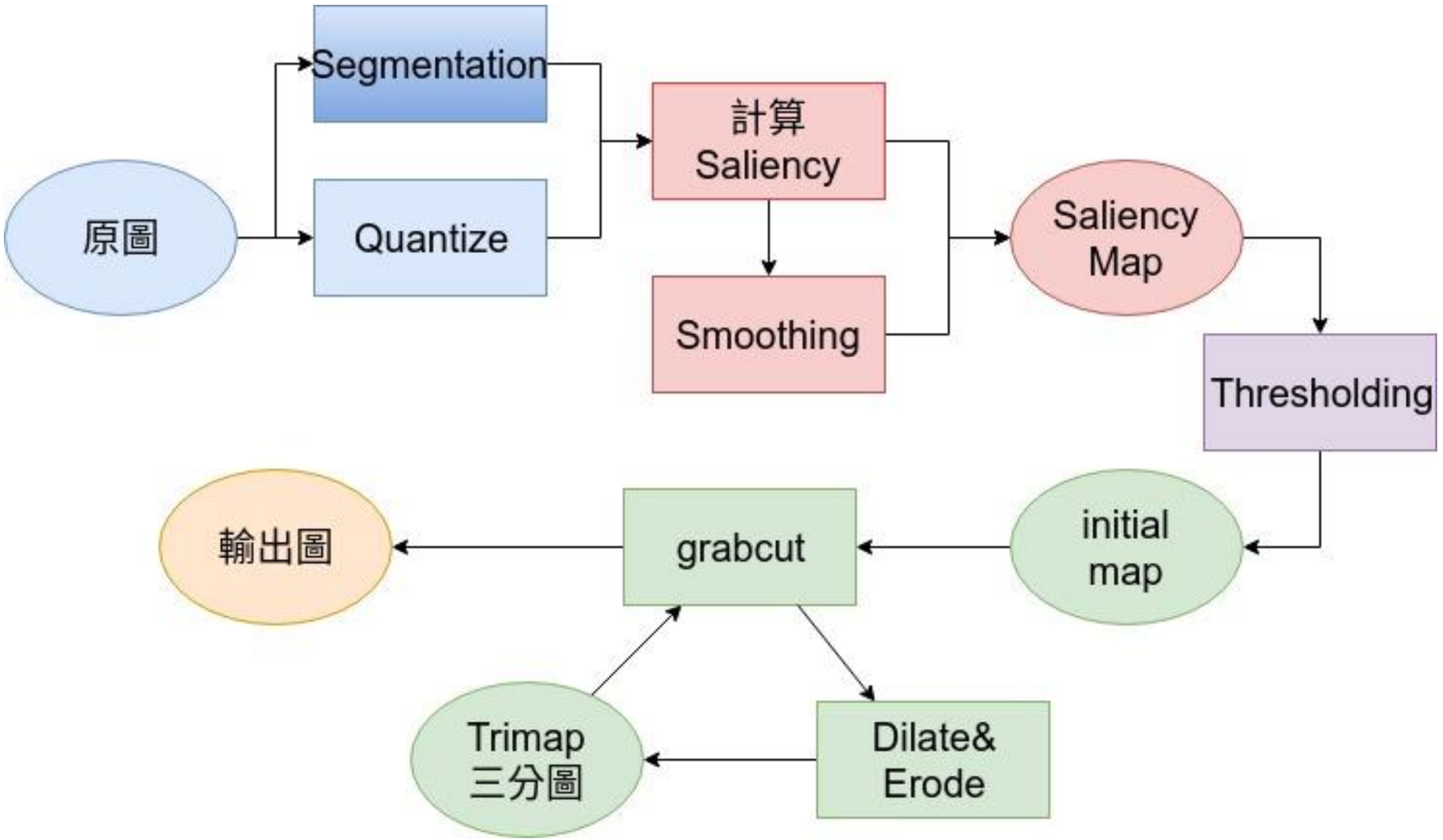


原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

計算saliency
生成initial mask



圖形預處理

使用grabcut多次迭代
輸出完成圖



原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

Segmentation

在計算saliency(顯著程度)之前，必須先將圖像切割為若干regions，這裡我們採用的是

Efficient Graph-Based Image Segmentation, by Felzenswalb and Huttenlocher

的經典圖像分割演算法。

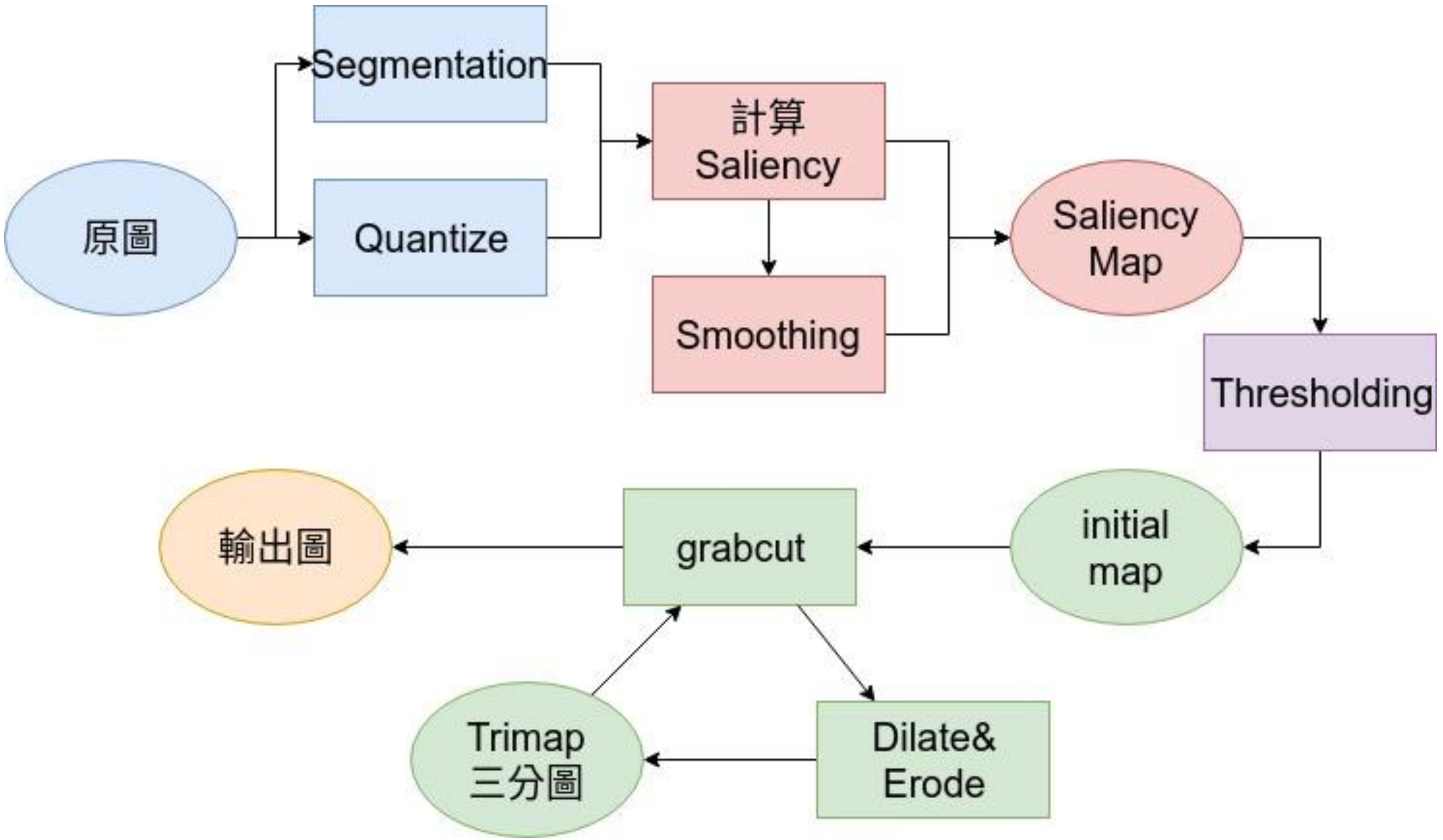
由於此部份並非我們實做論文的重點，

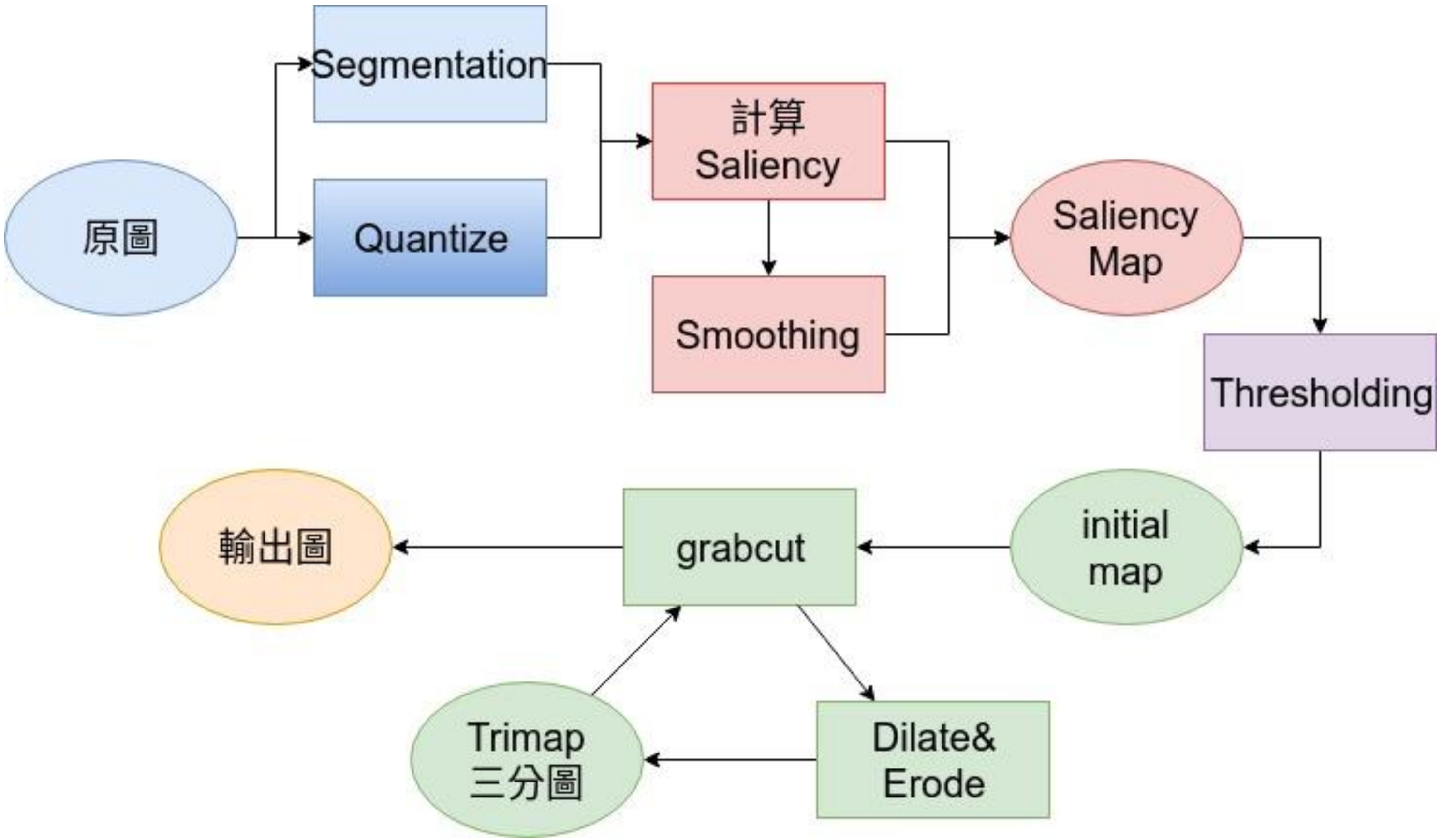
因此在segmentation的部份，我們使用了davidstutz所寫的

graph-based-image-segmentation，作為我們程式的預處理。

Segmentation

右圖為segmentation後的結果示意圖。


原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖



原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

Quantize

parameter: NUM_BITS, COVER_PERCENTAGE

減少顏色總數，保有足 夠的品質
減少顏色總數的目的是加速後面的運算

maskBit = 0xff << (8-NUM_BITS)
color = color & maskBit

最多有 $2^{3\times NUB\_BITS}$ 顏色

color number: 120033

color number: 1418

NUM_BITS = 4, quantize to 16 x 16 x 16

Quantize

parameter: NUM_BITS, COVER_PERCENTAGE

一張圖片的顏色，通常集中在某一些顏色

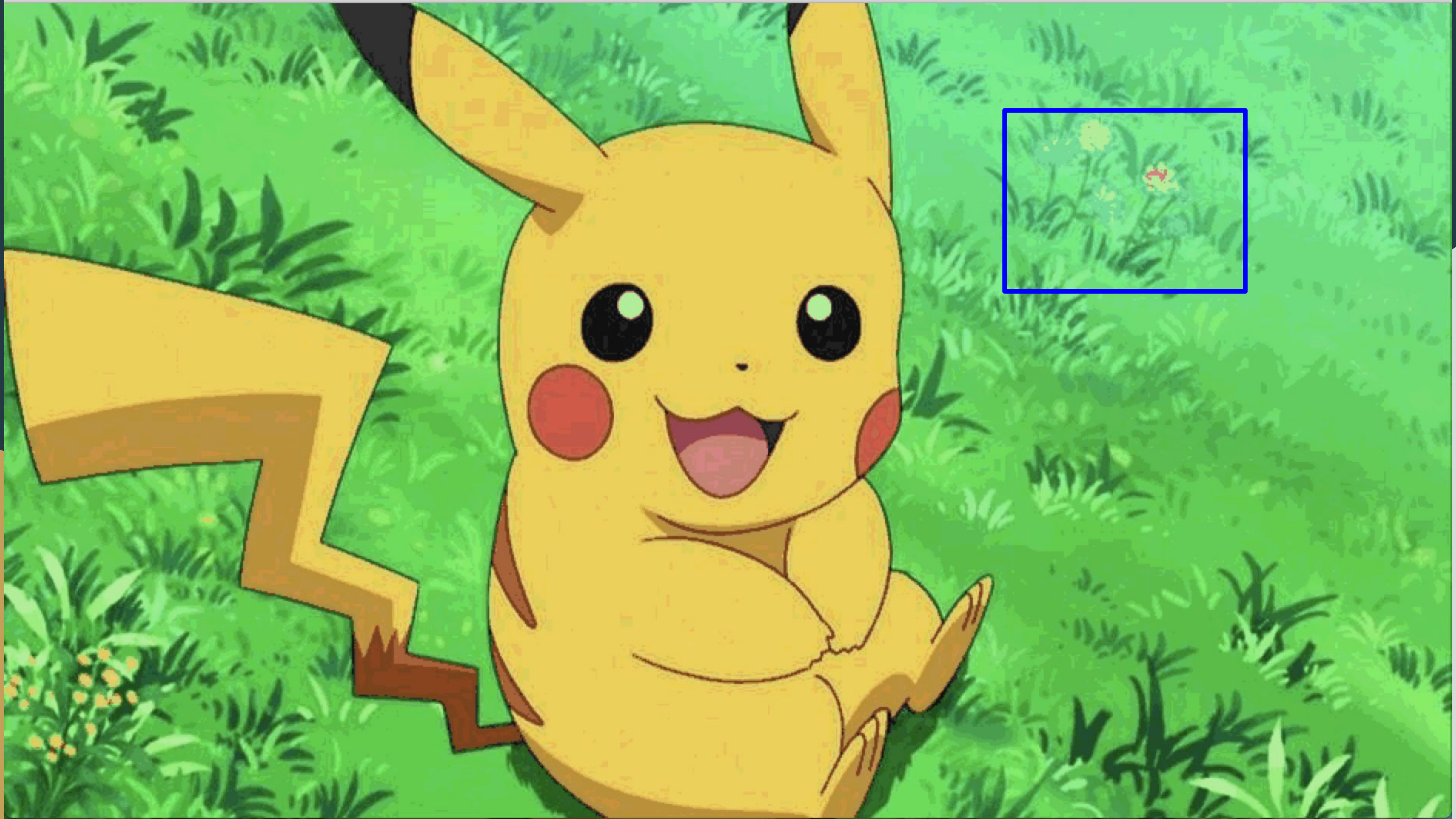
排序顏色出現頻率，從多到少開始選，直到覆蓋圖片超過COVER_PERCENTAGE

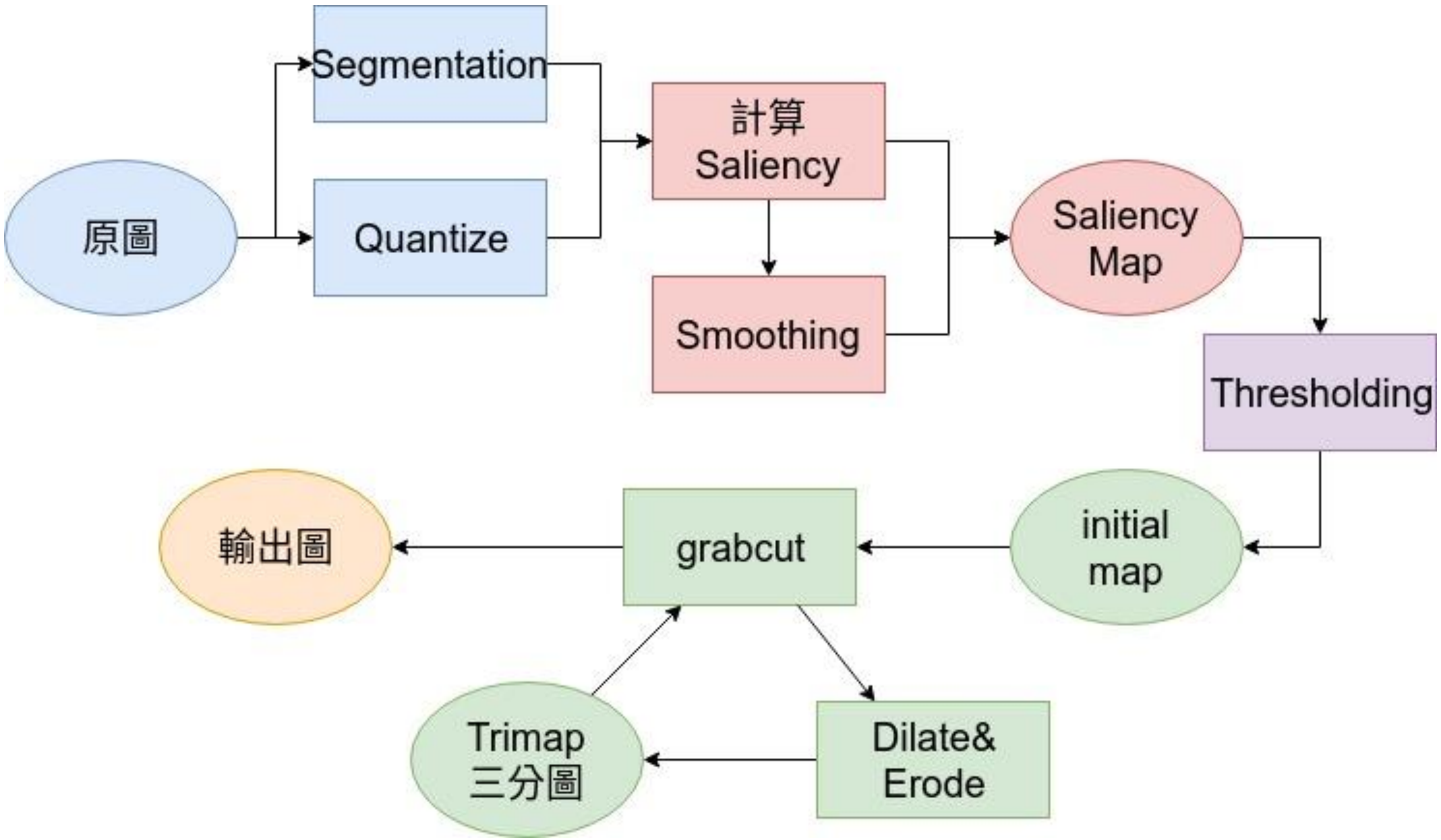
捨棄掉的顏色，從選進的顏色裡，挑最近的填補(l*a*b space)

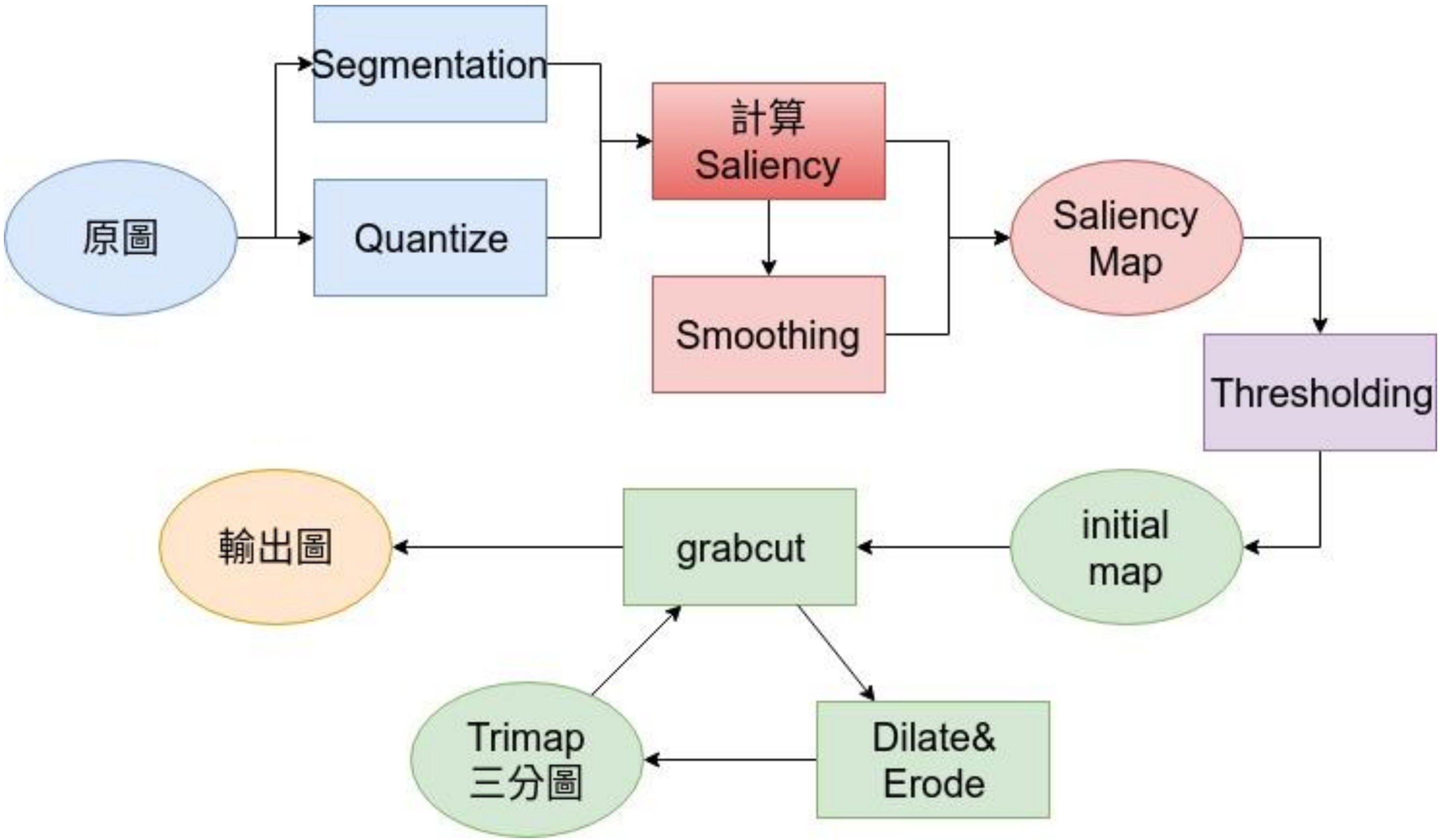
color: 1418

color: 172

COVER_PERCENTAGE: 90%




原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖


原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

Saliency Value

以區域為單位，計算分數，衡量區域在圖片中的突兀程度

(1) 區域間的顏色差異

(2) 不同區域顏色差異的權重

(3) 區域在圖片中的位置

Saliency Value

parameter: DELTA_S = 0.4

$$S(r_k) = w_s(r_k) \sum_{r_k \neq r_i} e^{\frac{D_s(r_k - r_i)}{-\sigma_s^2}} w(r_i) D_r(r_k, r_i)$$

$$S(r_k) = w_s(r_k) \sum_{r_k \neq r_i} e^{\frac{D_s(r_k - r_i)}{-\sigma_s^2}} w(r_i) D_r(r_k, r_i)$$

$$D_r(r_k, r_i) = \sum_{i=1}^{n1} \sum_{j=1}^{n2} f(c_{1,i}) f(c_{2.j}) D(c_{1,i} - c_{2,j})$$

$f(c_{k,i})$是在區域 k 中全部n_k種顏色的第 i 個顏色$c_{k,i}$在整張圖片出現的機率

$w(r_i)$ 區域中的pixel數

WHY we need QUANTIZATION!

$$D_r(r_k, r_i) = \sum_{i=1}^{n1} \sum_{j=1}^{n2} f(c_{1,i}) f(c_{2.j}) D(c_{1,i} - c_{2,j})$$

$$S(r_k) = w_s(r_k) \sum_{r_k \neq r_i} e^{\frac{D_s(r_k - r_i)}{-\sigma_s^2}} w(r_i) D_r(r_k, r_i)$$

$d_s(r_k, r_i)$: 是兩區域質心的距離

σ_s 控制質心距離的影響力

距離較近的顏色差異，代表變化很大，權重較重！

$$S(r_k) = \boxed{w_s(r_k)} \sum_{r_k \neq r_i} e^{\frac{D_s(r_k - r_i)}{-\sigma_s^2}} w(r_i) D_r(r_k, r_i)$$

$$w_s(r_k) = \exp\left(-9{d_k}^2\right)$$

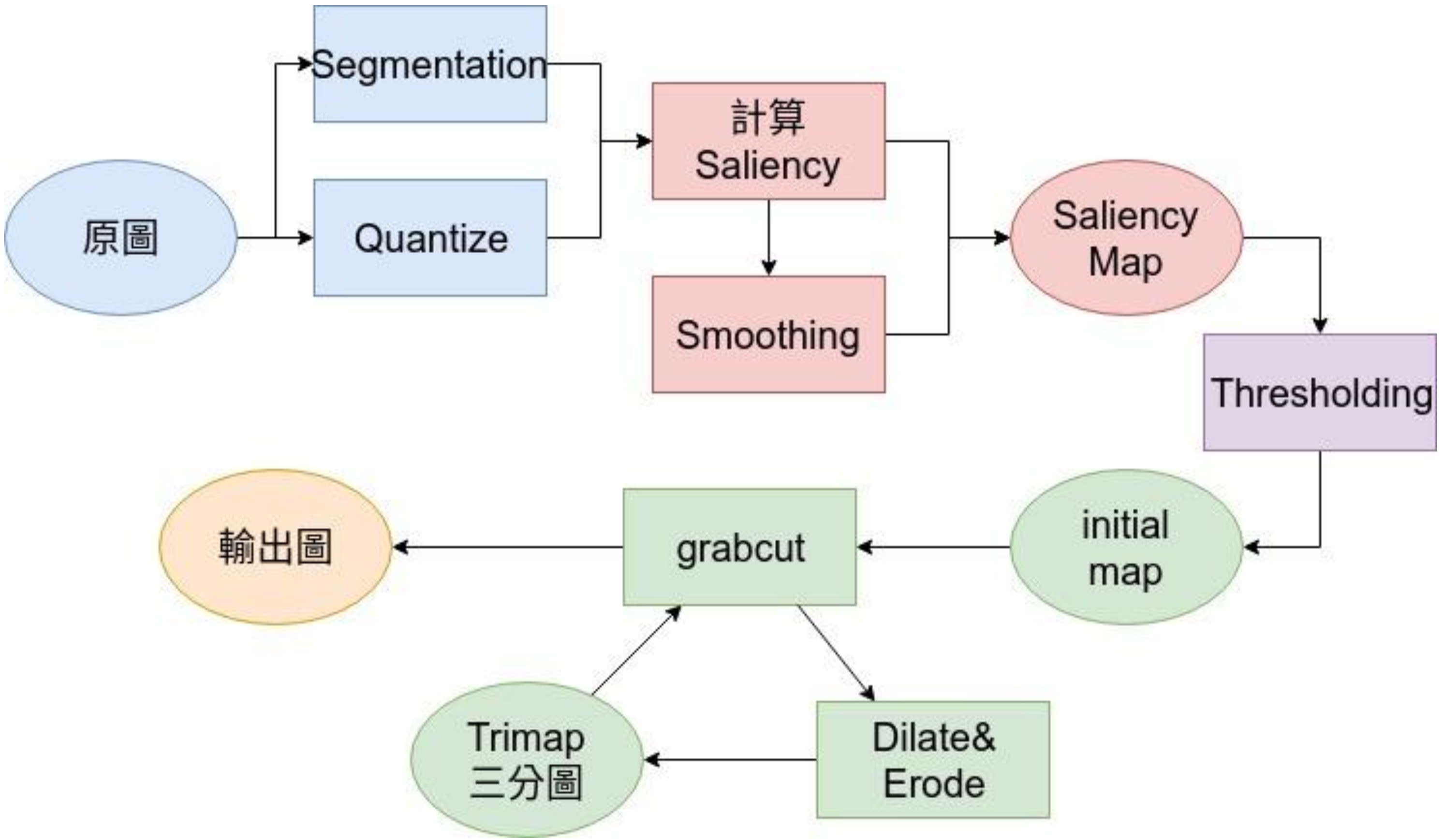
d_k 是區域中與圖片中心的平均距離

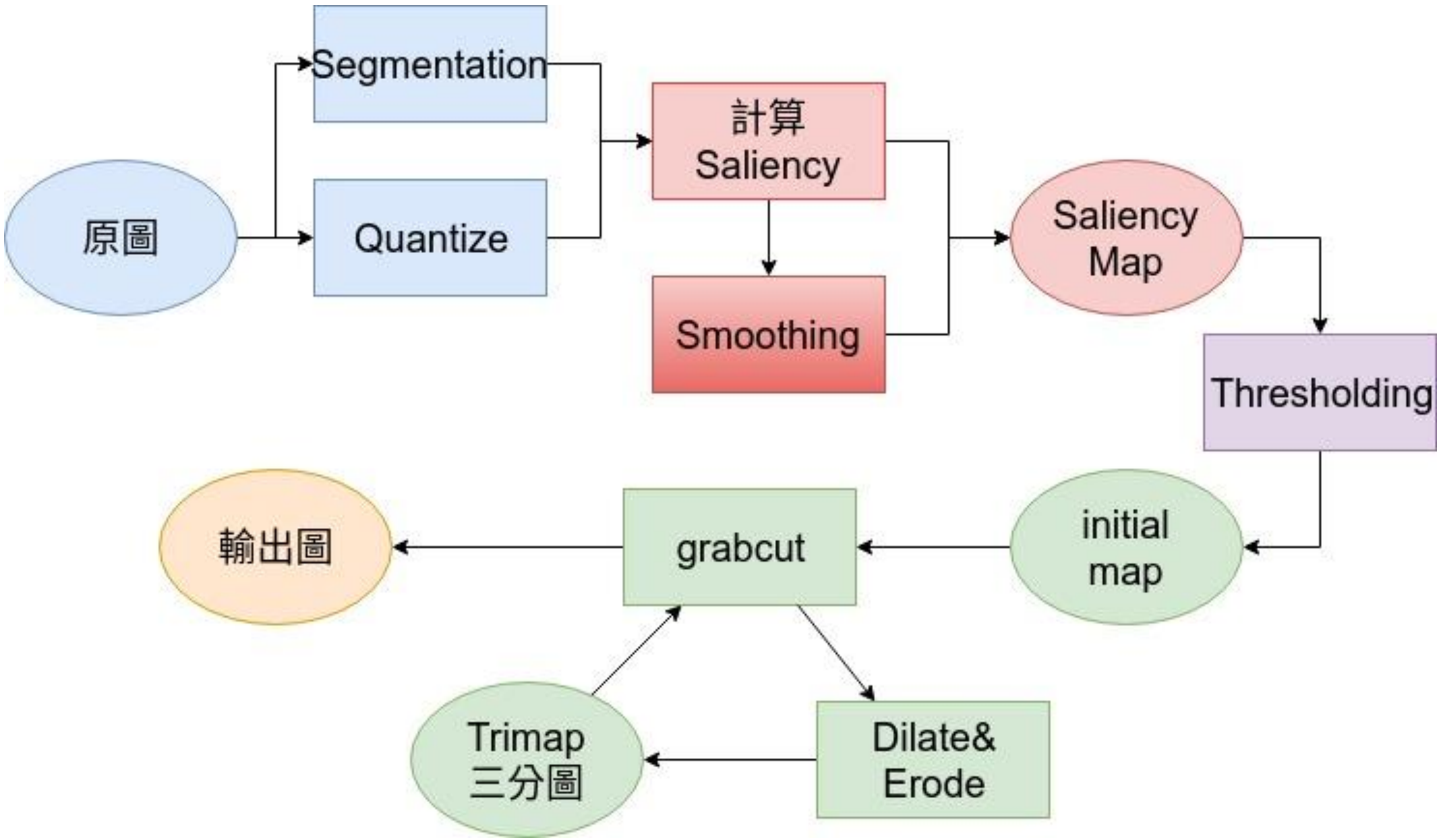
越靠近中心的，越有可能是主體，權重越重！

RGB
quantization

v.s

LAB
color distance


原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖


原圖
Segmentation
Quantize
計算 Saliency
Smoothing
Saliency Map
Thresholding
initial map
grabcut
輸出圖
Dilate& Erode
Trimap 三分圖

color smooth

目標：解決quantization artifact，讓相近的顏色有相近的saliency value

Color smooth

parameter: SMOOTH_RATIO

$$S'(c) = \frac{1}{(m-1)T}\sum_{1}^{m}(T - D(c, c_i))S(c_i)$$

m為所有n種顏色中，最靠近c的m種顏色（lab space）

$$m = n * SMOOTH\_RATIO$$

$$S'(c) = \frac{1}{(m-1)T}\sum_{1}^{m}(T - D(c, c_i))S(c_i)$$

$$T = \sum_{1}^{m} D(c, c_i)$$

$$T - D(c, c_i)$$

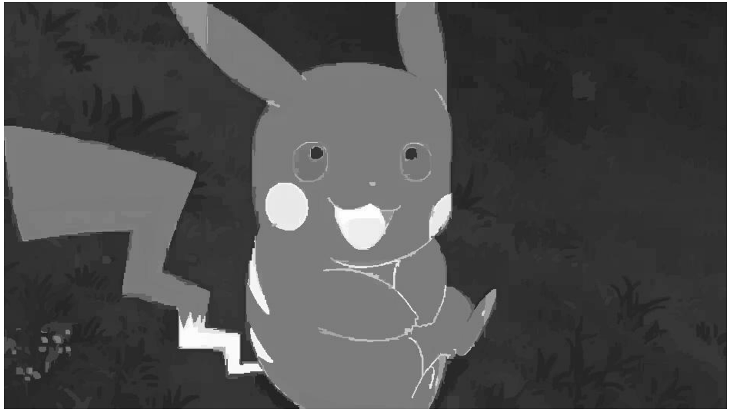
差異越大的，權重越輕

$$S'(c) = \frac{1}{(m-1)T}\sum_{1}^{m}(T - D(c, c_i))S(c_i)$$

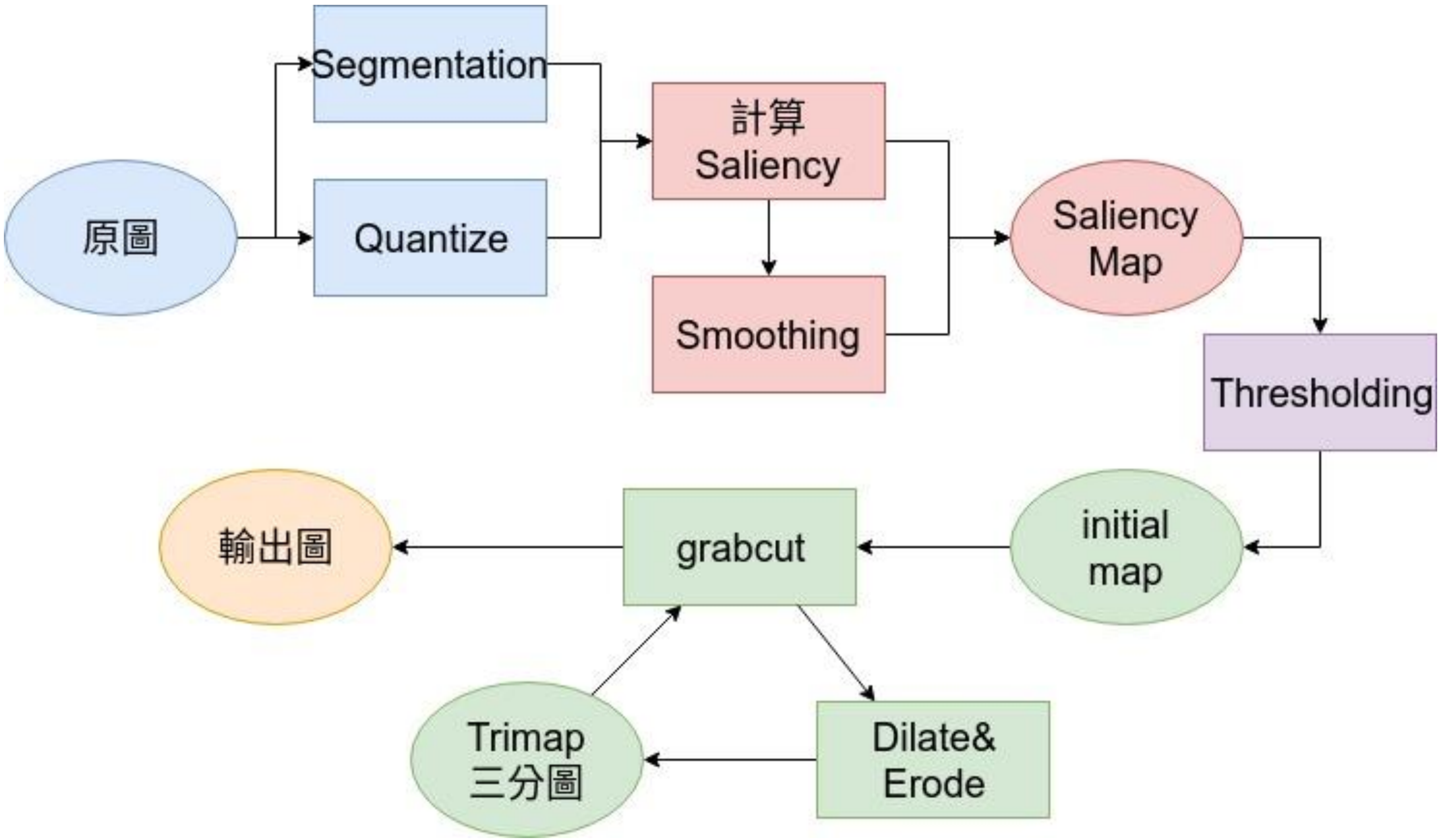
$$\sum_{1}^{m} T - D(c, c_i) = (m-1)T$$

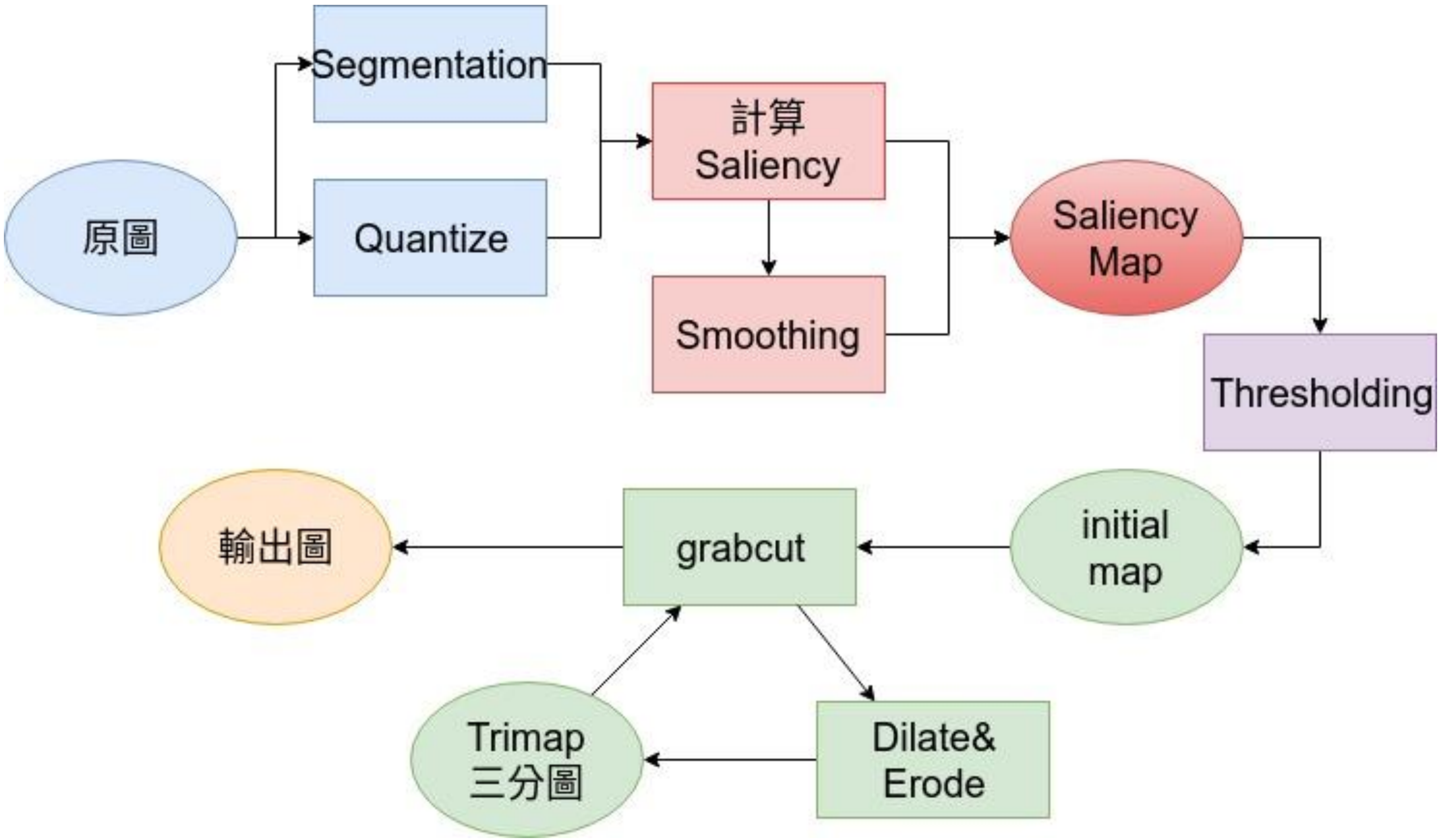
Normalized term

Original saliency map



Color smooth


原圖
Segmentation
Quantize
計算 Saliency
Smoothing
Saliency Map
Thresholding
initial map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖



原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

why mix

parameter: S_RATIO

color smooth後，顏色相近的區域saliency value相近

然而，卻也失去位置的資訊

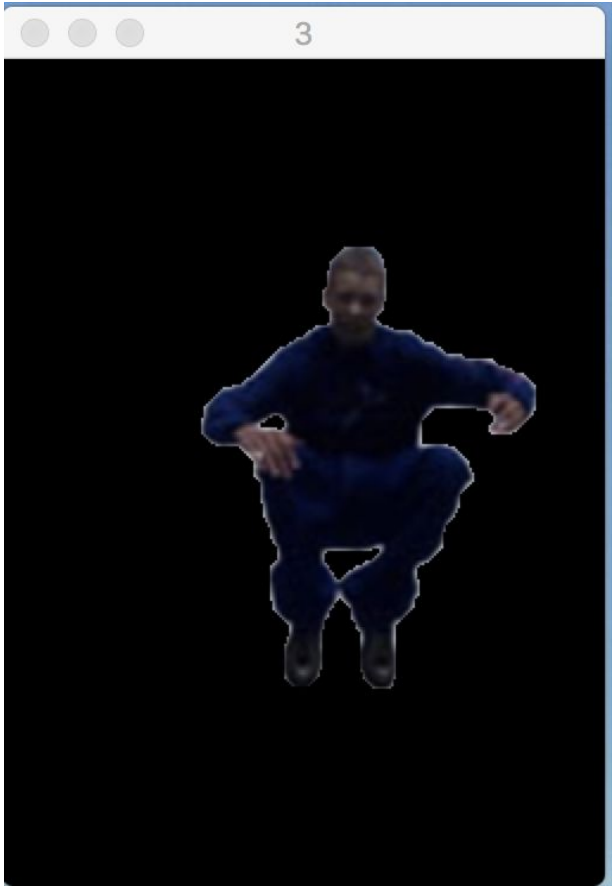
why mix

parameter: S_RATIO

$$Salincy = \underbrace{S_{RATIO} \times oldSaliency}_{\text{location information}} + \underbrace{(1 - S_{RATIO}) \times smoothSaliency}_{\text{reduce artifact, similar color similar value}}$$

Original

Mix


3

Time performance

Method	IT [17]	AIM [22]	IM [23]	MSS [24]	SEG [25]	SeR [26]	SUN [27]	SWD [28]	CB [35]
Time (s)	0.246	4.288	0.991	0.106	4.921	1.019	1.116	0.100	5.568
Code Type	Matlab	Matlab	Matlab	C++	Matlab	Matlab	Matlab	Matlab	Matlab & C
Method	GB [29]	SR [30]	FT [33]	AC [31]	CA [32]	LC [34]	HC	RC	
Time (s)	1.614	0.064	0.102	0.109	53.1	0.018	0.019	0.254	
Code Type	Matlab	Matlab	C++	Matlab	Matlab	C++	C++	C++	

Fig. 7. Average time taken to compute a saliency map for images in the MSRA10K database (most have resolution 400×300). We use parallel computing environment for all Matlab functions for efficient computation.

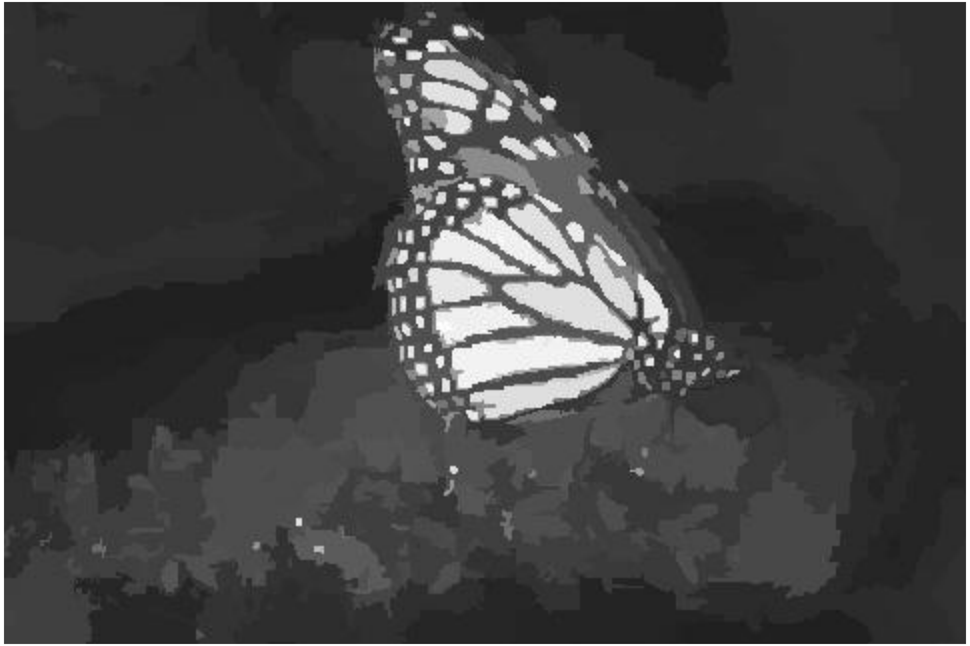

9K G

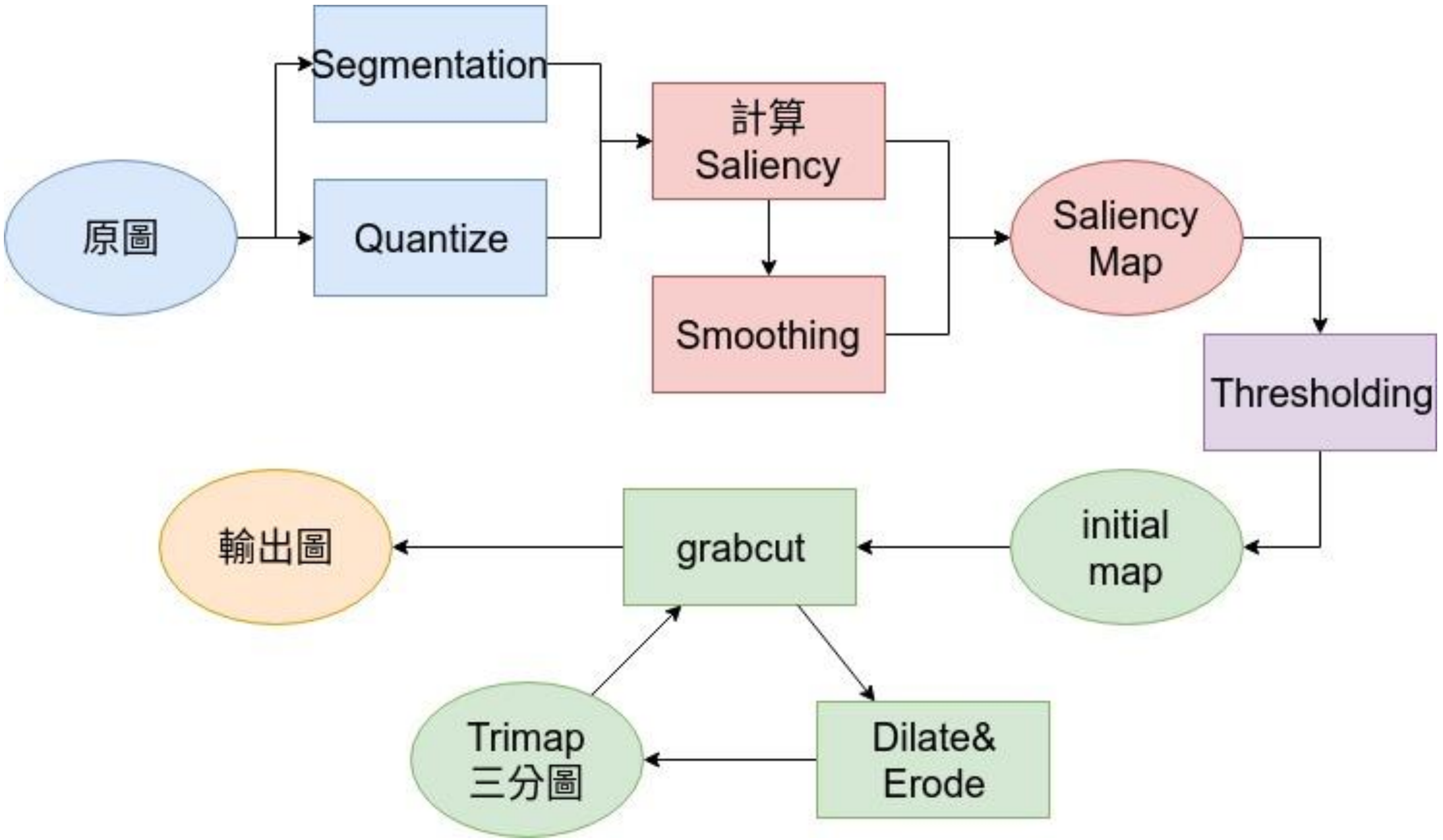
Time performance

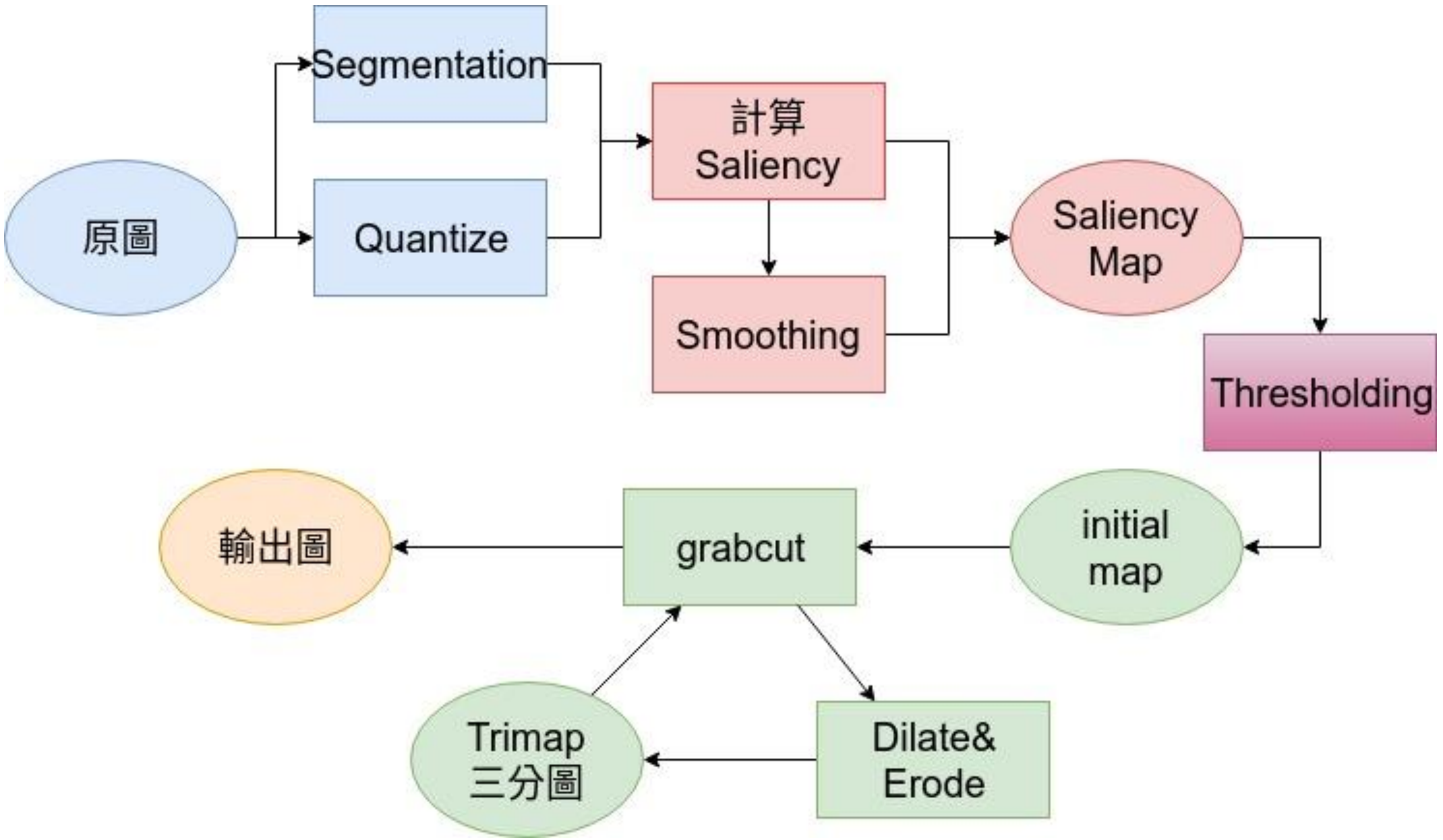
```
caiminghongde-MacBook-Pro:main tall154215452$ time ./DIP ../test/3063.jpg
It's a 321 * 481 image

real    0m0.708s
user    0m0.663s
sys     0m0.164s
```

Time performance

```
caiminghongde-MacBook-Pro:main tall154215452$ time ./DIP ../test/35049.jpg
It's a 321 * 481 image

real	0m0.956s
user	0m0.906s
sys	0m0.171s
```

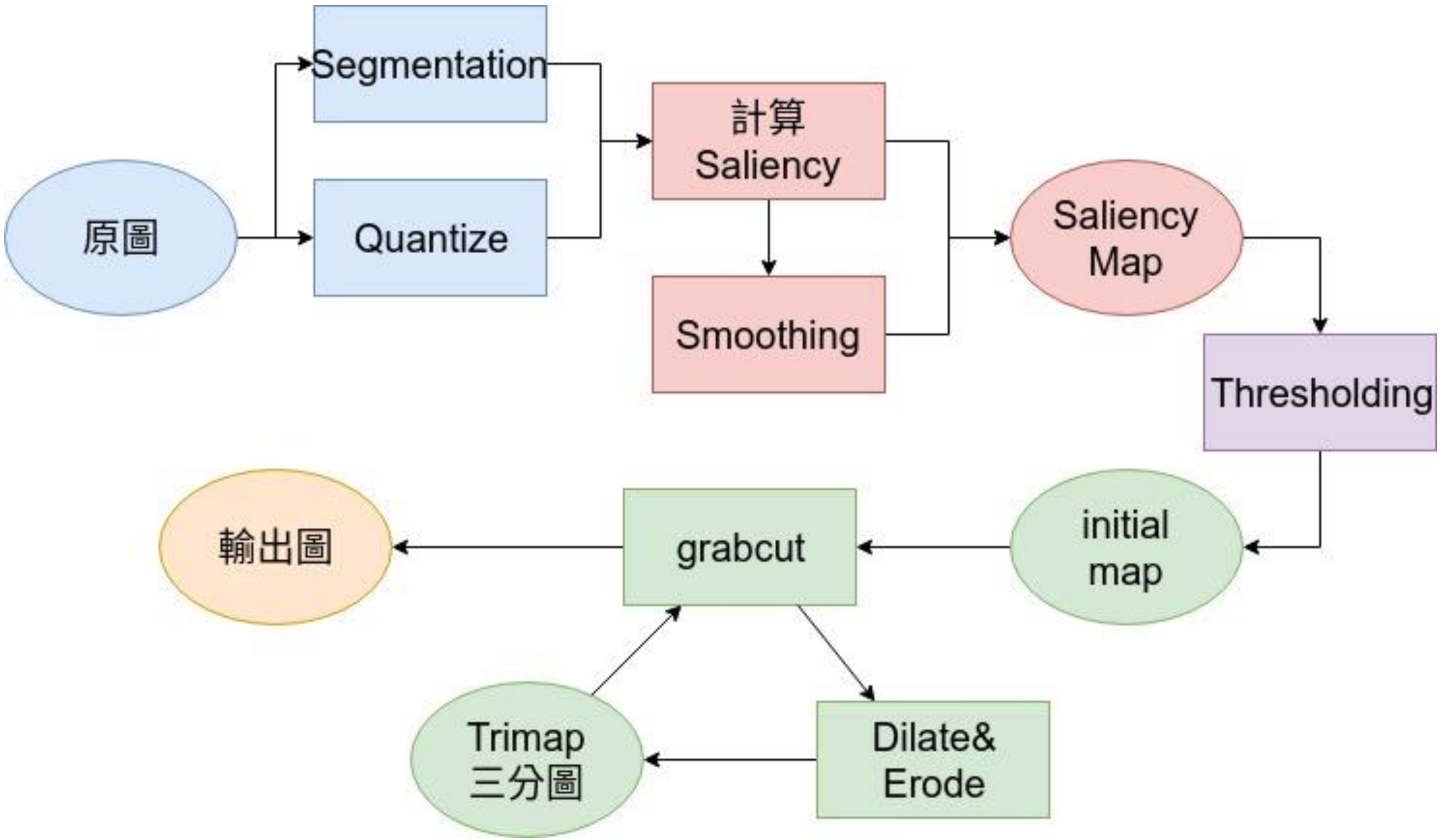



原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

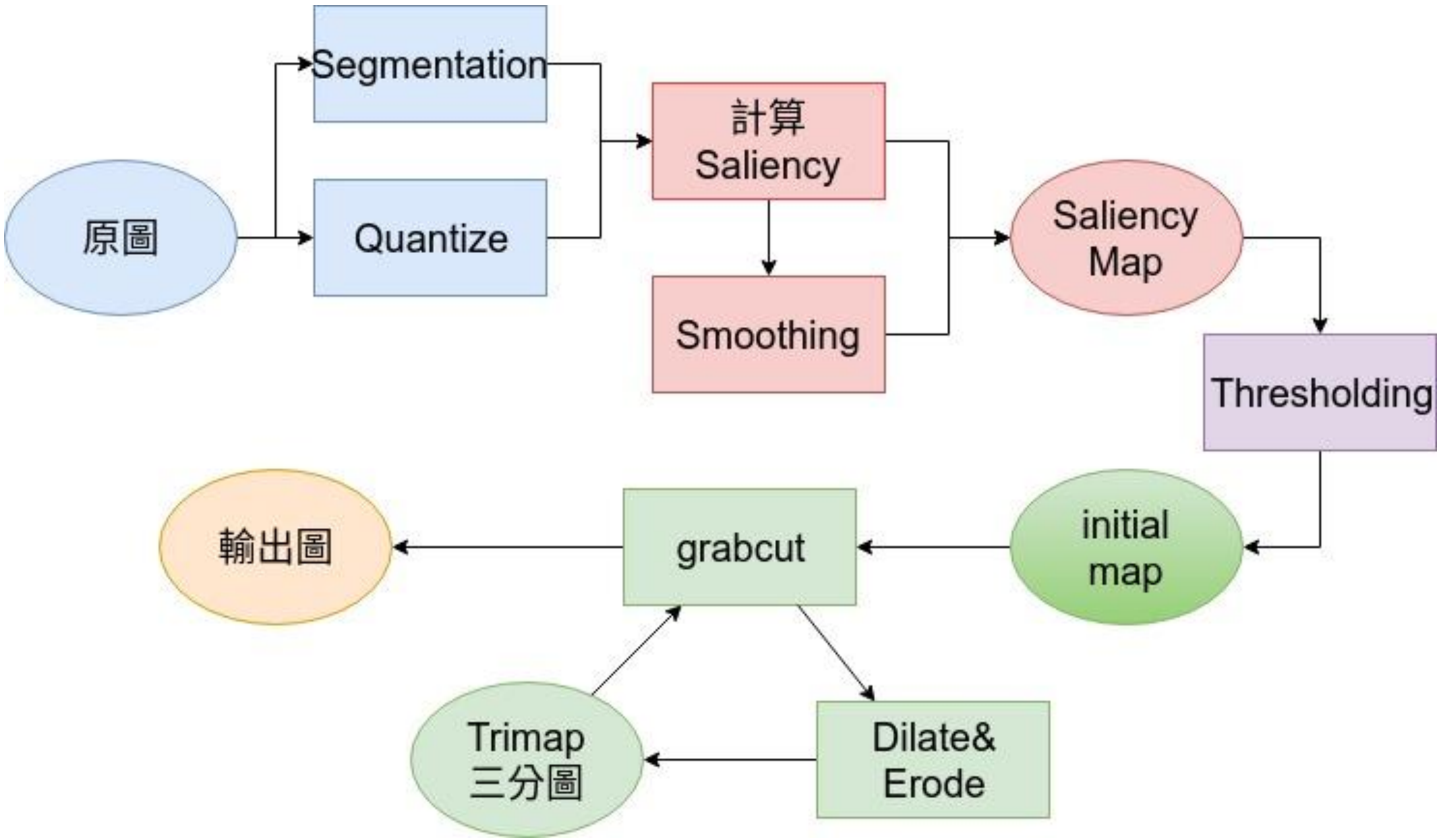

原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

Threshold

paper use Threshold = 70



原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Trimap
三分圖
Dilate&
Erode

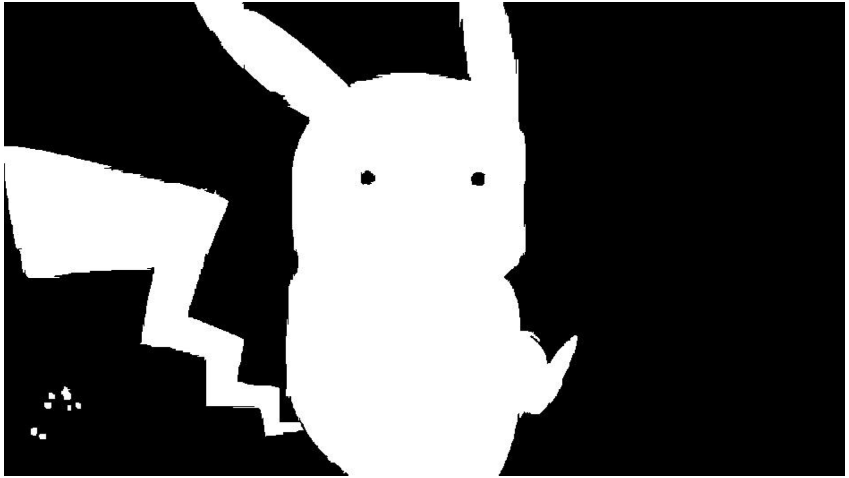

原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

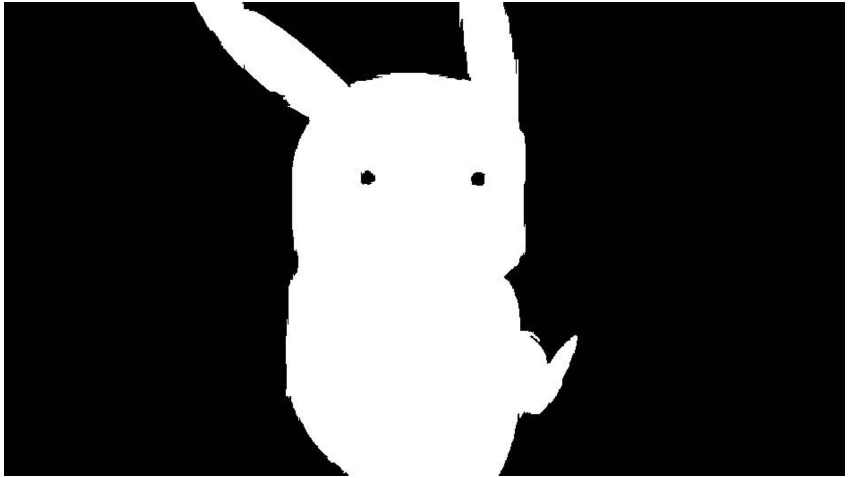
initial map

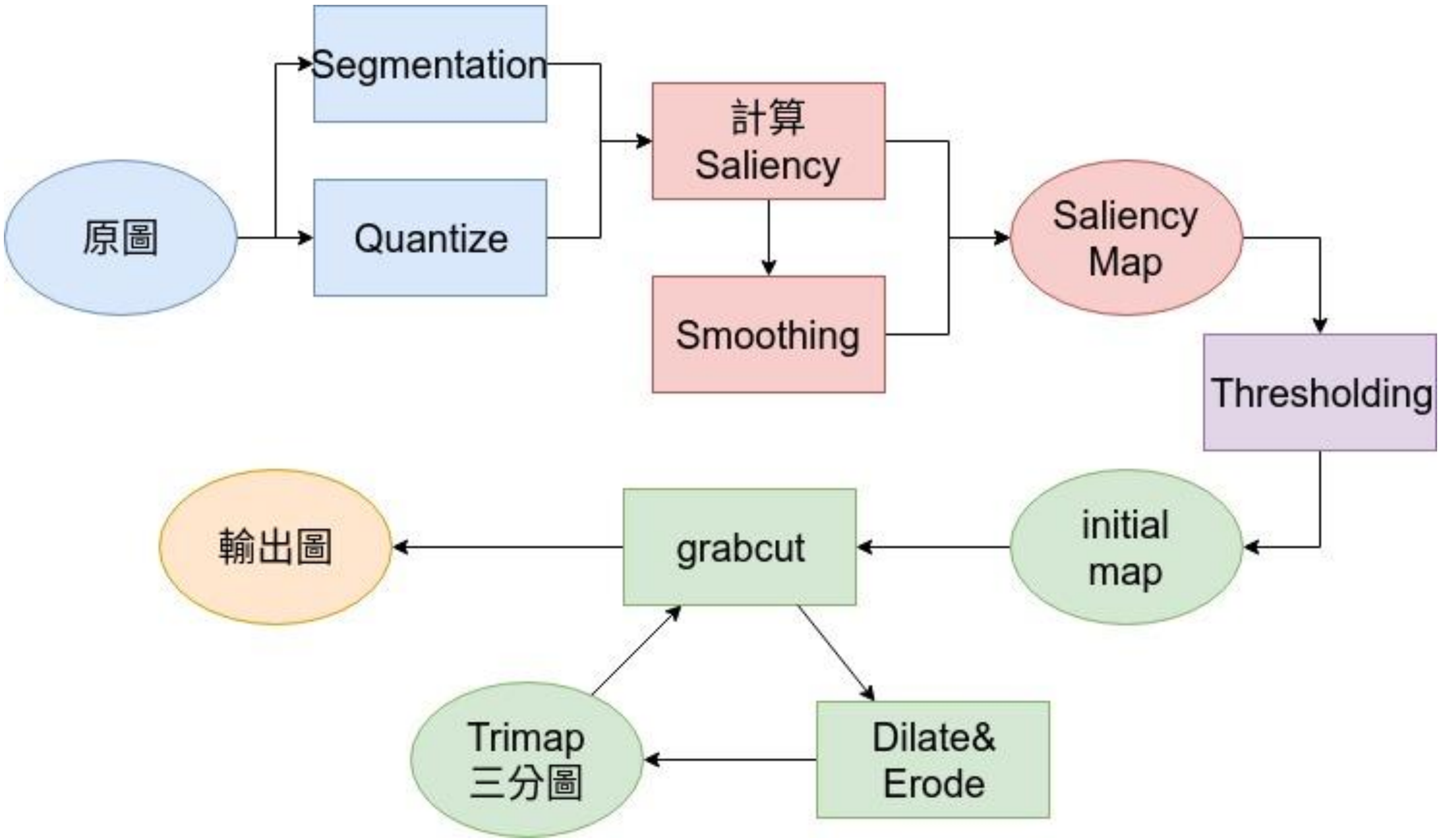
- 在thresholding後，我們會得到初始化後的binary map，用來標示物體所在。
- 一般而言，foreground object都會是相鄰的，因此我們找出binary map中最大的region並保留，其他則予以刪除。

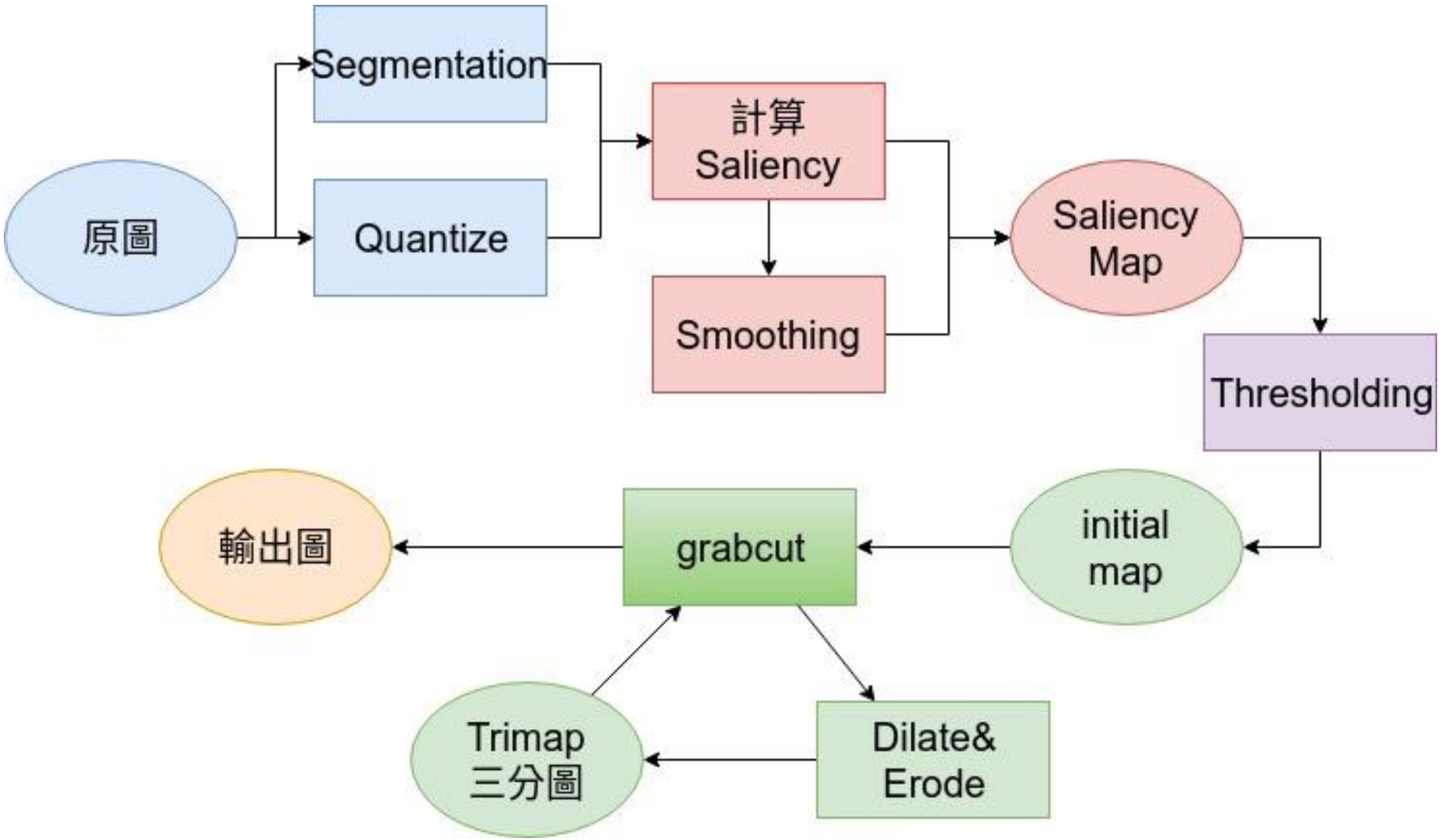
initial map

在這張圖中，左下角的果實被誤認為是foreground，而在只保留最大區域的處理後，成功消除了左下角的果實。至於尾巴？別緊張....後面我們可以把它救回來。






原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖



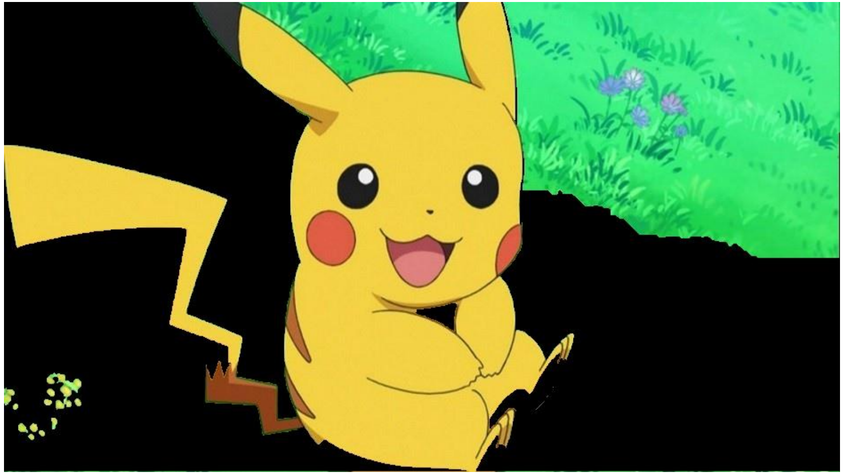
原圖
Segmentation
Quantize
計算 Saliency
Smoothing
Saliency Map
Thresholding
initial map
grabcut
輸出圖
Dilate& Erode
Trimap 三分圖

Grabcut

- 在Graph cut基礎上改進的一種圖像分割演算法
- 需要事先指定foreground、background的範圍，才能得到較好的效果
- 我們所使用的是OpenCV提供的內建函式 : grabCut()

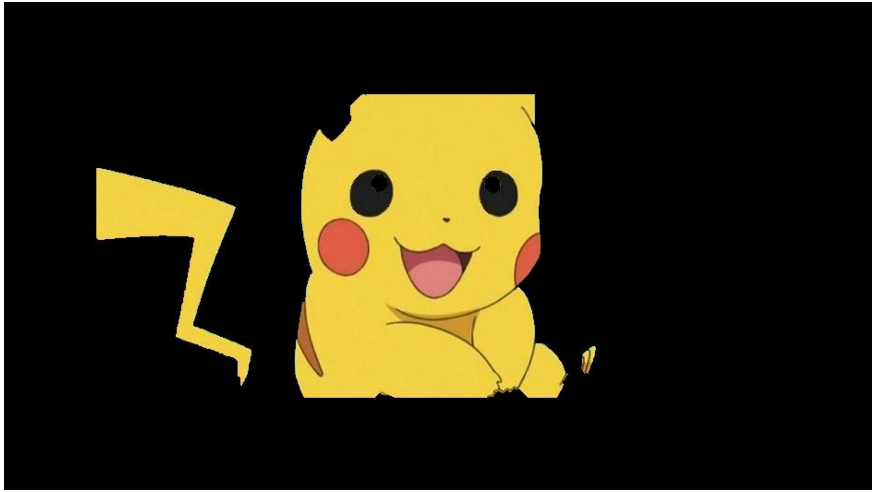
Grabcut

不進行foreground標記，直接套用Grabcut 函式的結果。



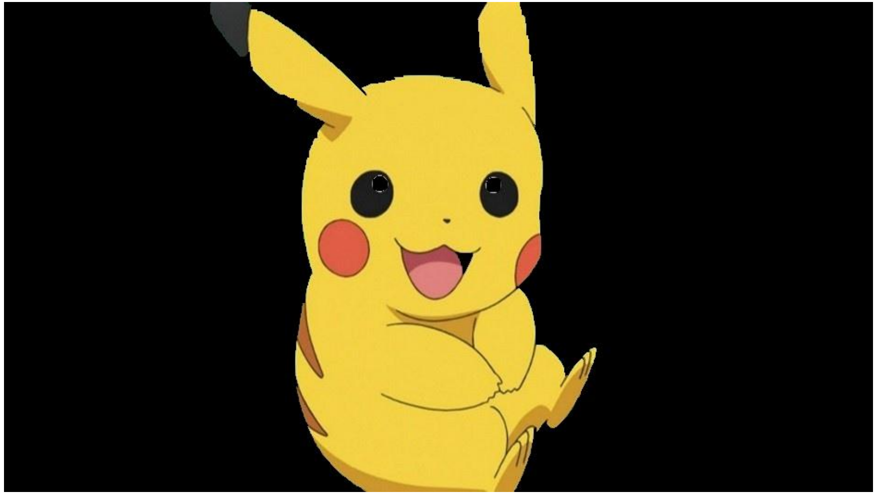
Grabcut

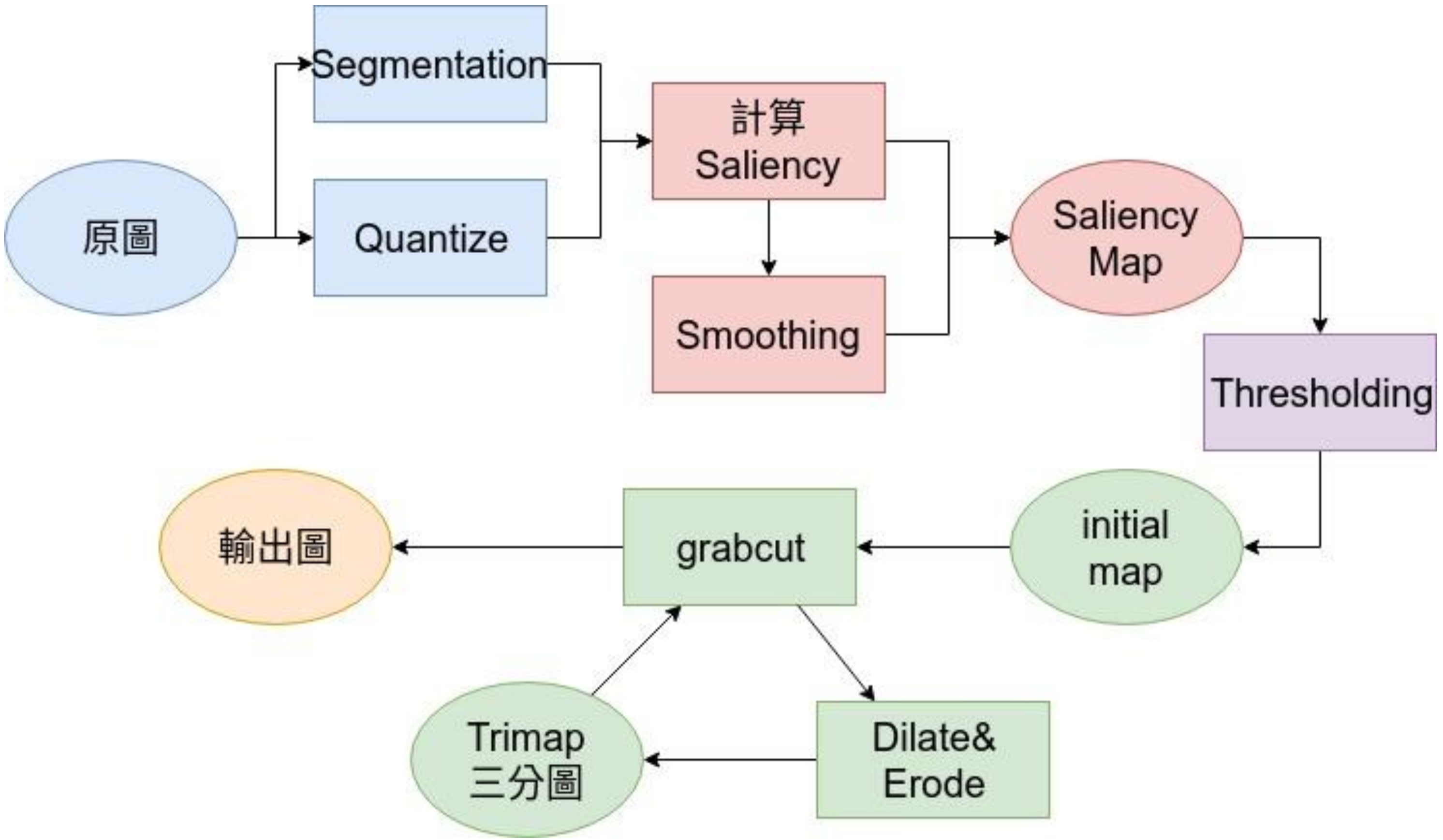
進行foreground標記後，grabcut便能得到較好的區域效果。

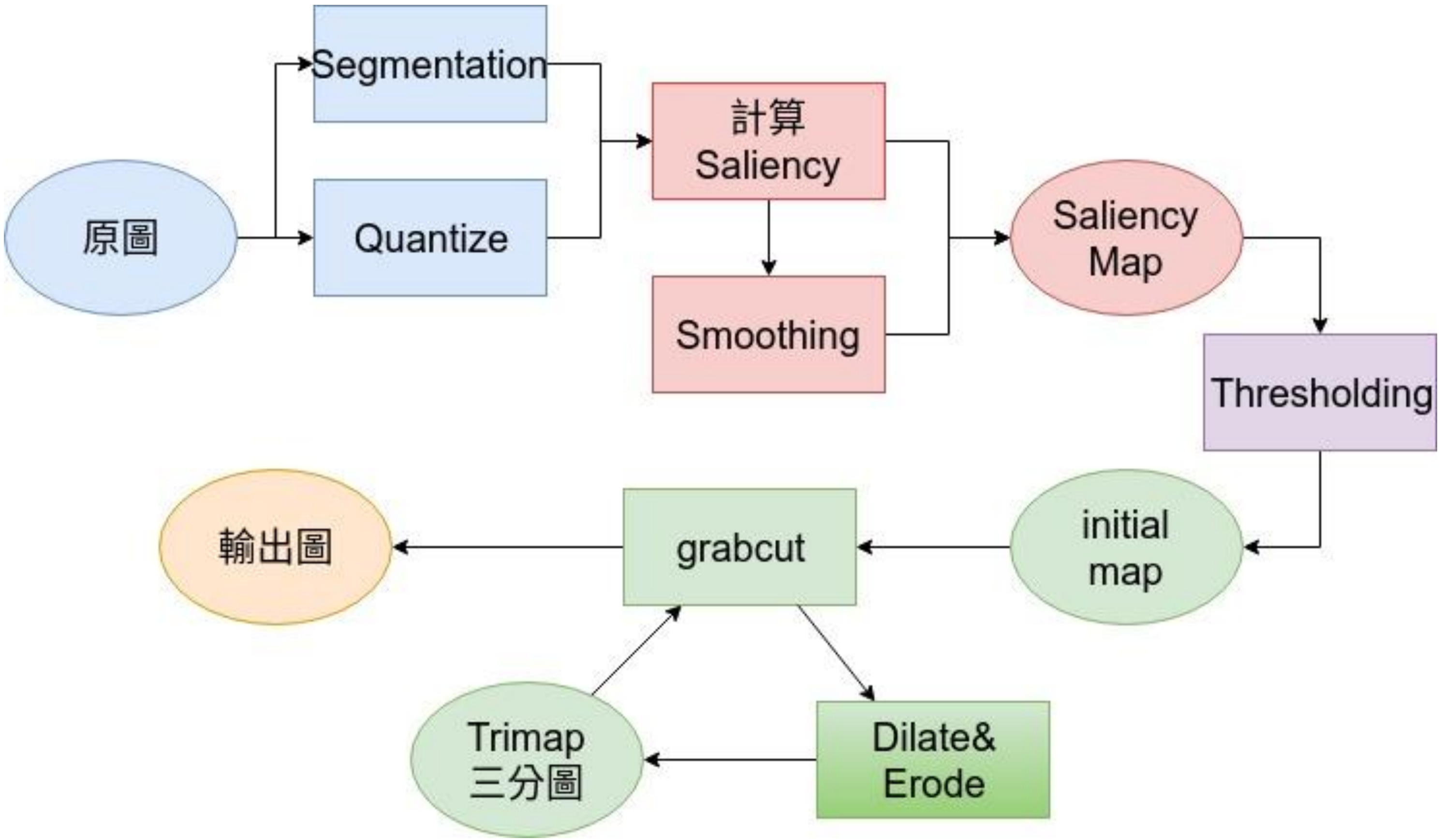


Grabcut

這是經過計算saliency後，標記出foreground區域並且grabcut後的結果



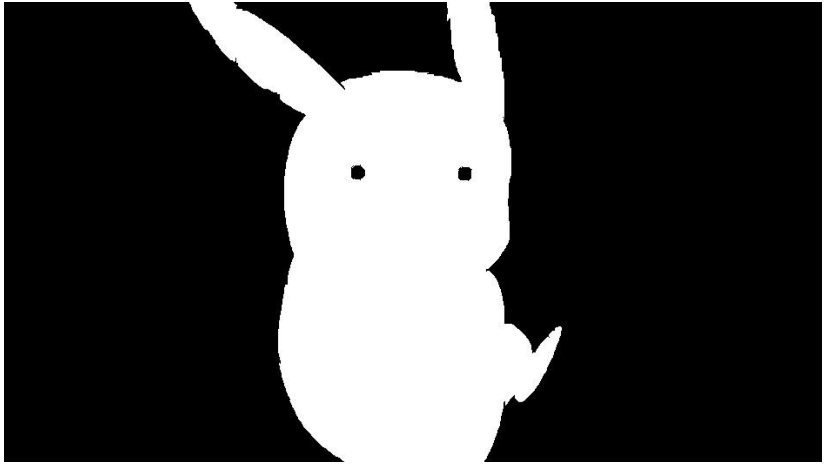

原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

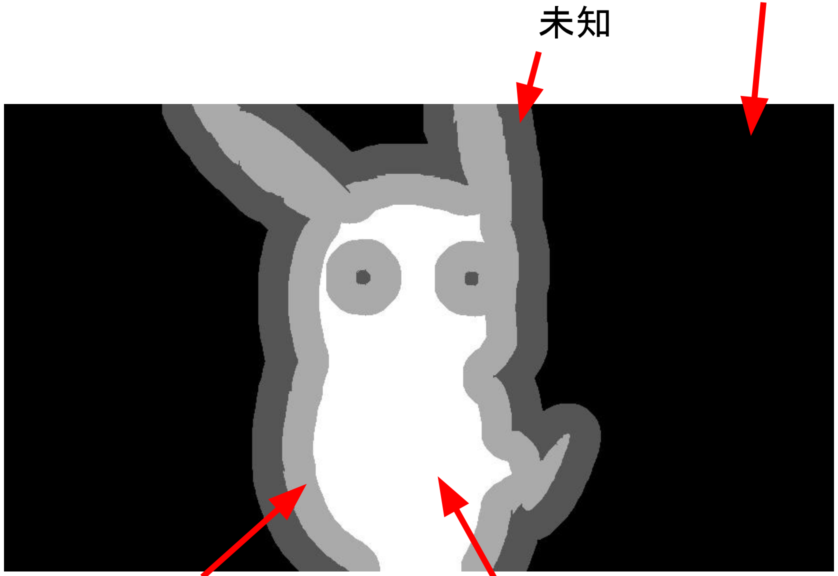

原圖
Segmentation
Quantize
計算 Saliency
Smoothing
Saliency Map
Thresholding
initial map
grabcut
輸出圖
Dilate& Erode
Trimap 三分圖

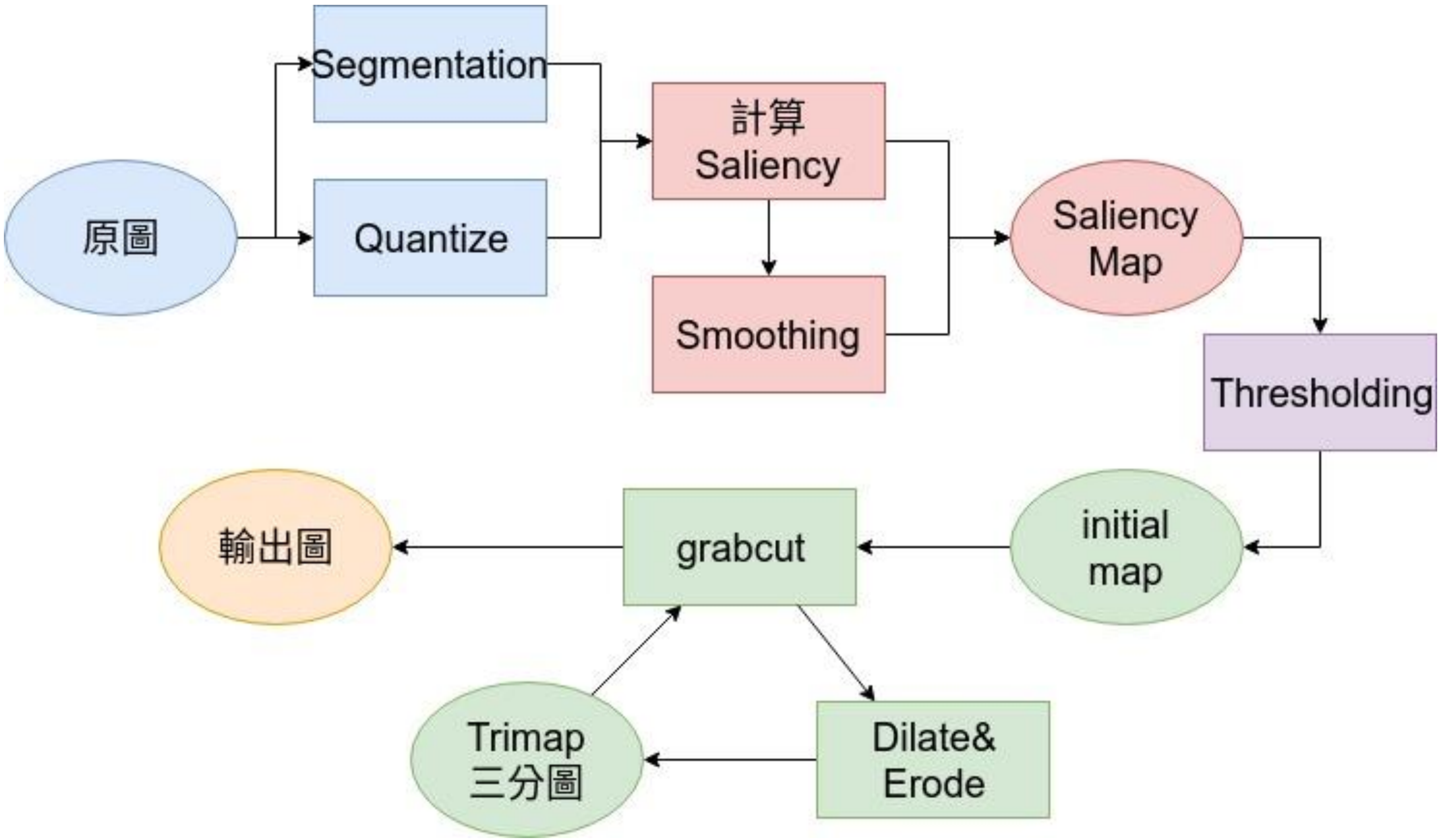
Dilate&Erode

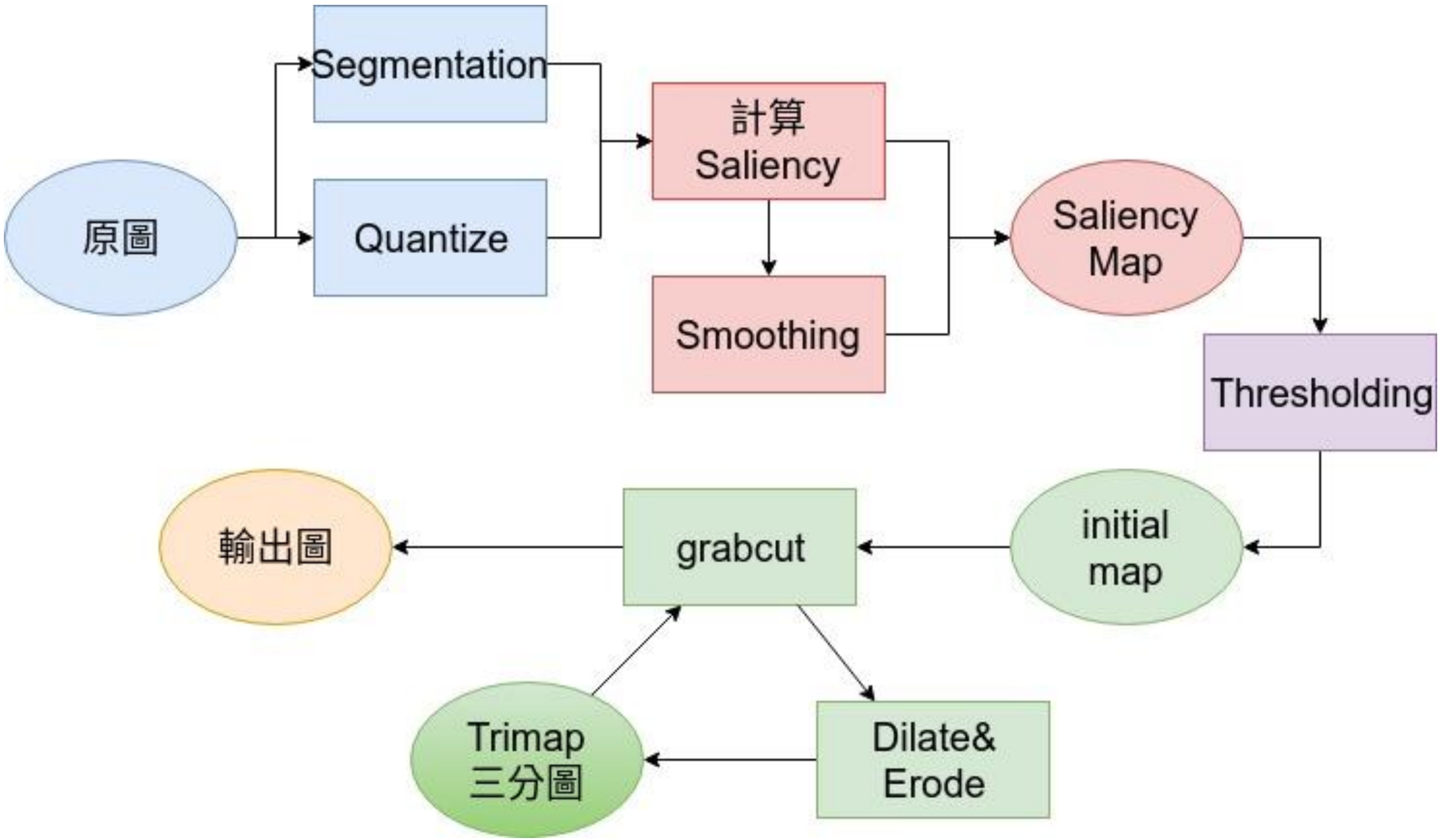
對於grabcut出來的結果，我們重新對它分別進行Dilate及Erode，

以利於待會將圖形三分化。



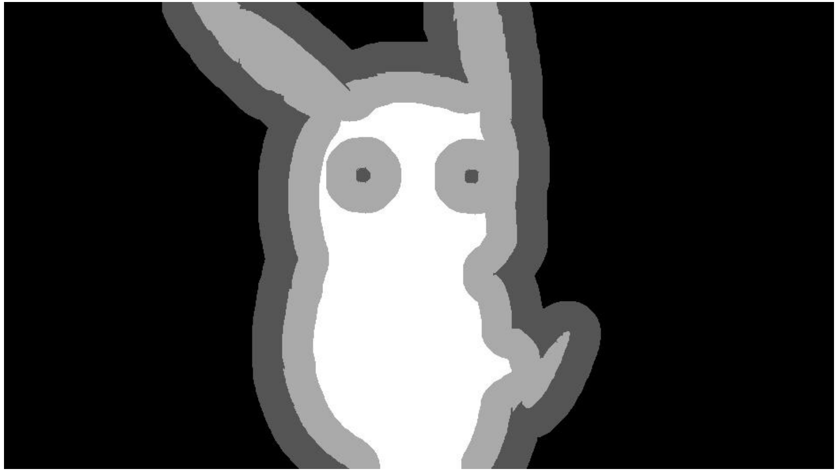


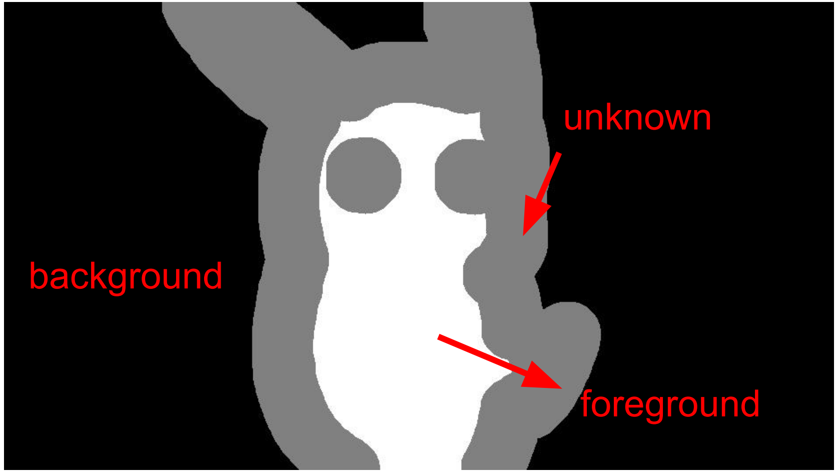

原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖


原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Trimap
三分圖
Dilate&
Erode

Trimap三分圖

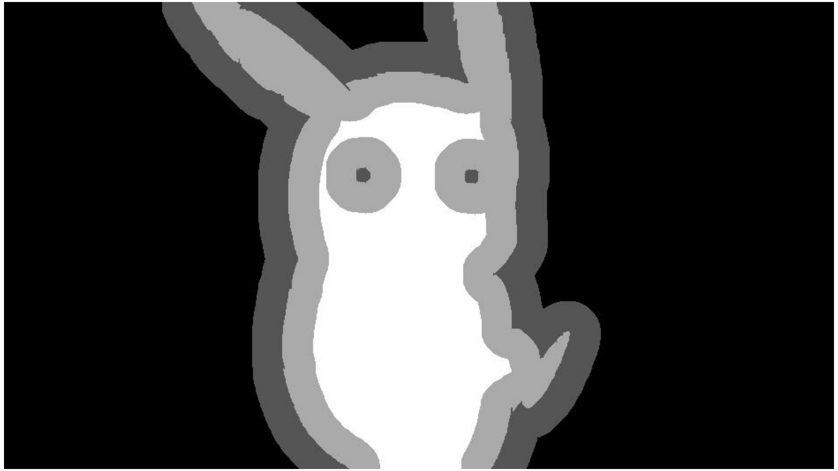
利用剛剛Dilate&Erode後的結果將圖三分化，將前後結果都是1的pixel設為foreground，0的pixel設為background，有變動的pixel則設為unknown。

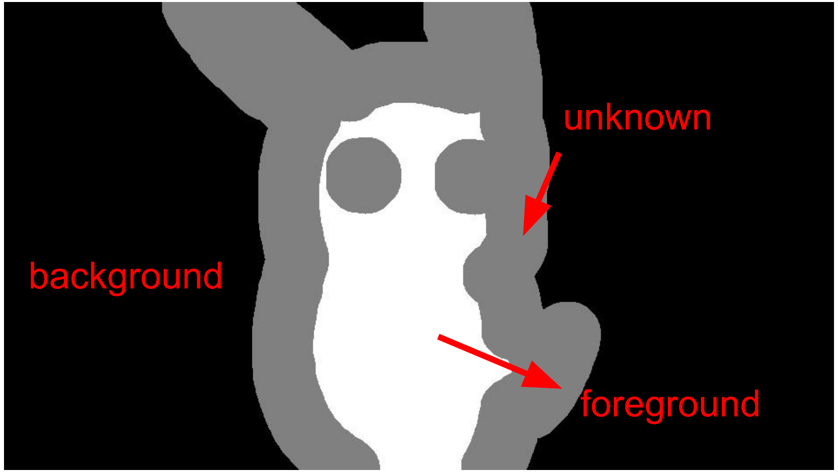


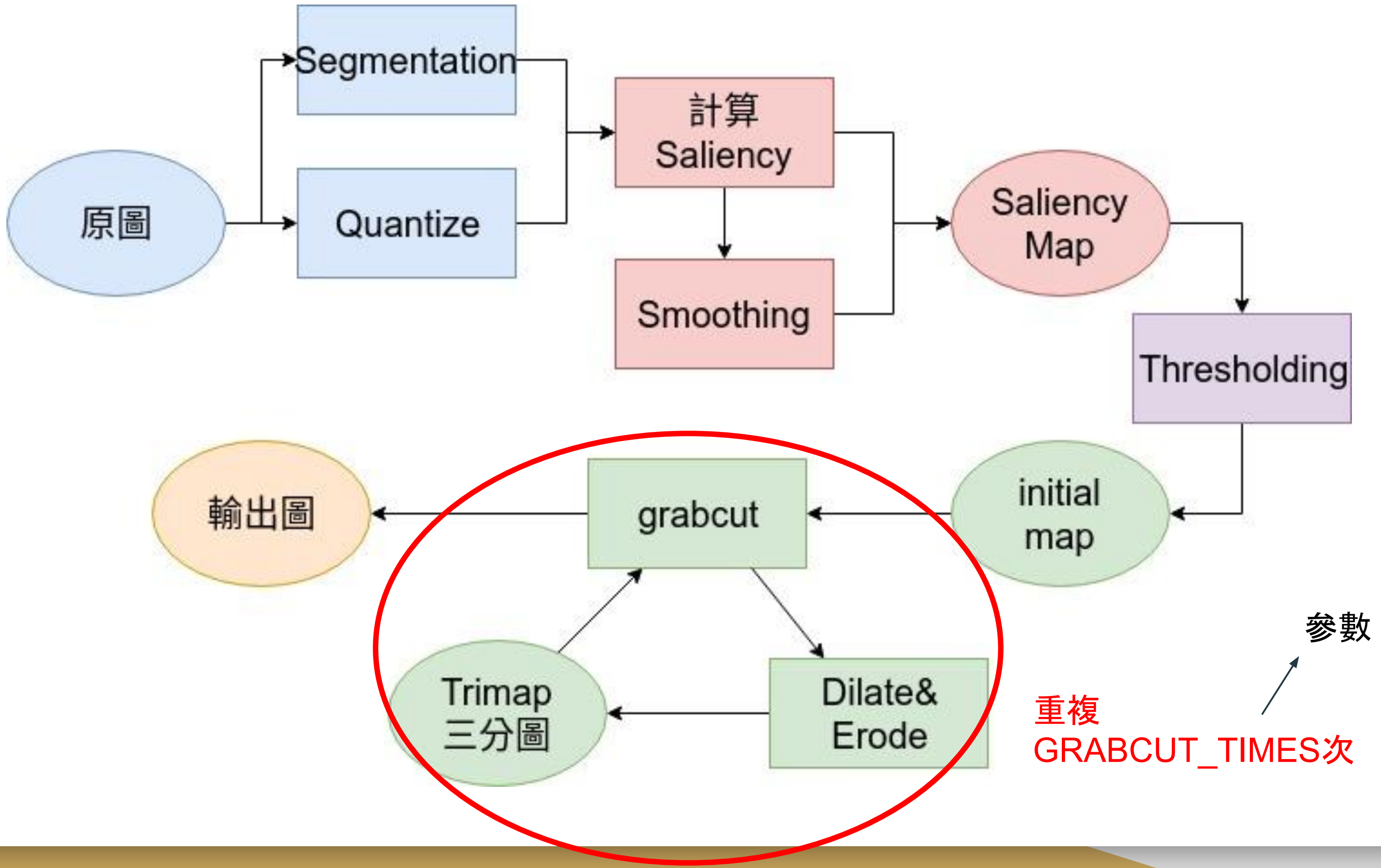


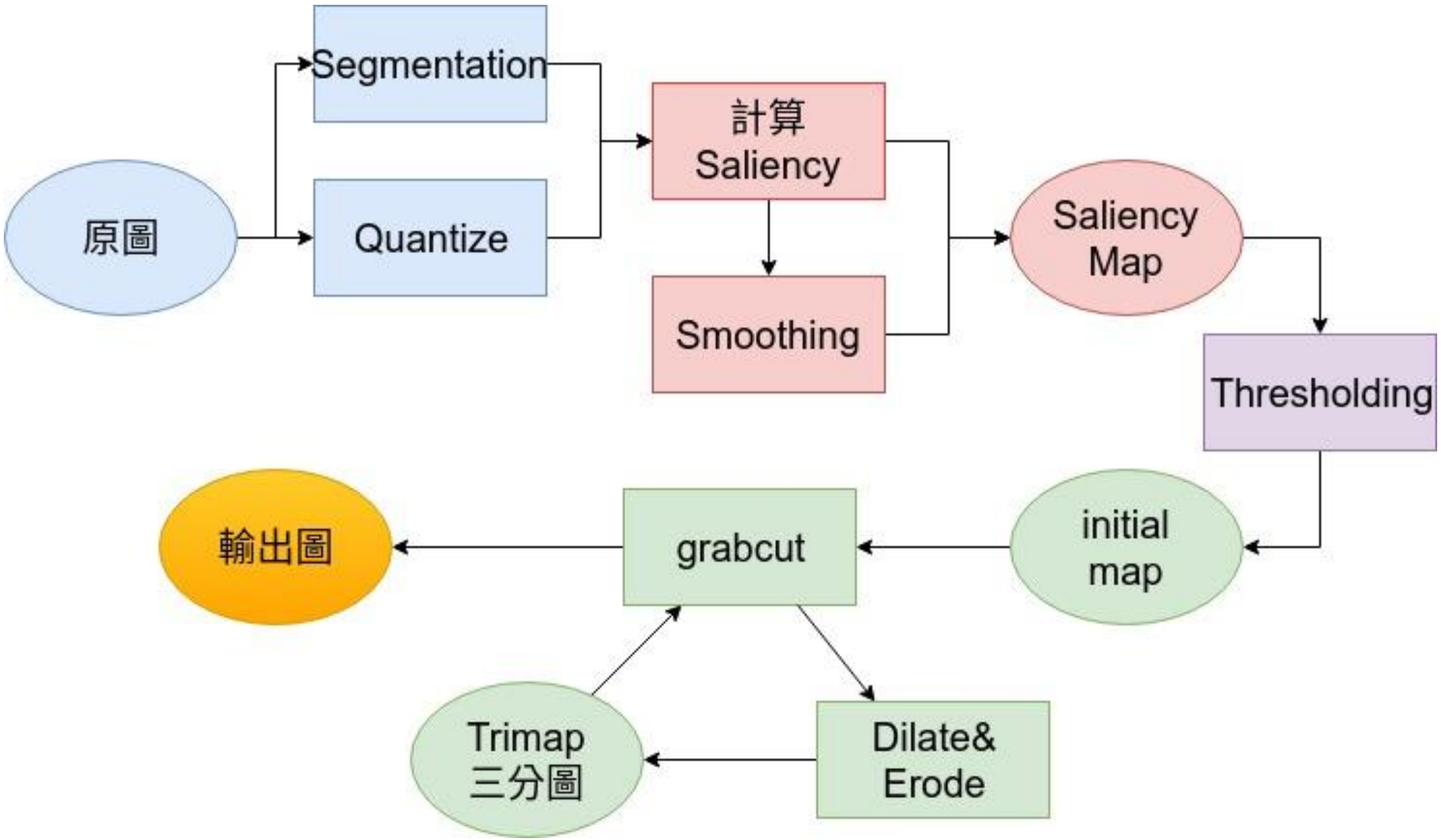
Trimap三分圖

將圖三分化後，重新送進grabcut做運算。



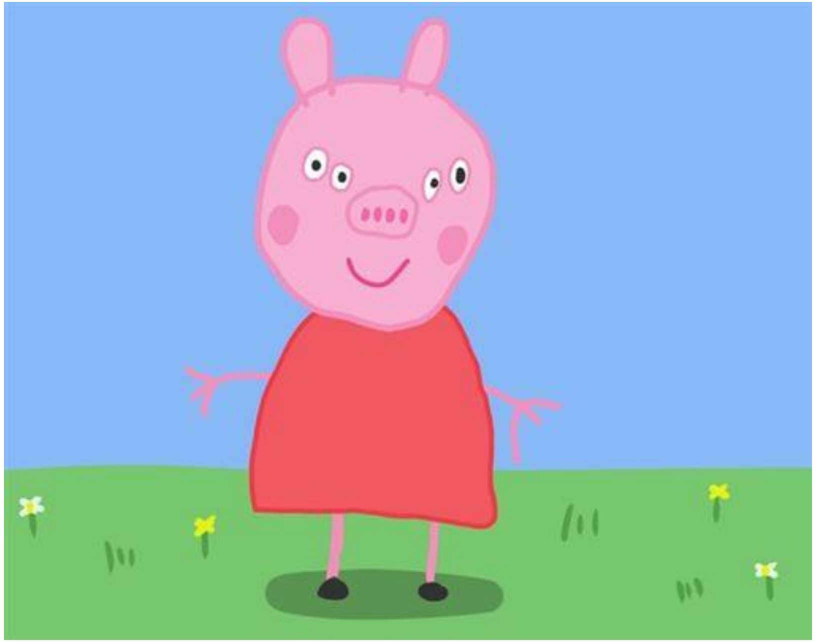


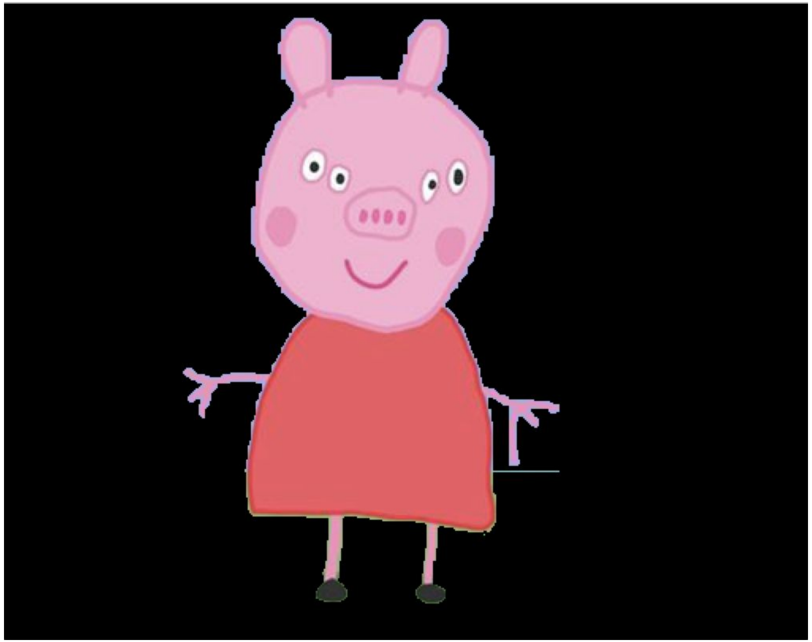

原圖
Segmentation
Quantize
計算 Saliency
Smoothing
Saliency Map
Thresholding
initial map
grabcut
Dilate& Erode
Trimap 三分圖
輸出圖
參數
重複
GRABCUT_TIMES次


原圖
Segmentation
Quantize
計算
Saliency
Smoothing
Saliency
Map
Thresholding
initial
map
grabcut
輸出圖
Dilate&
Erode
Trimap
三分圖

Experiment result

Experiment result





Experiment result

Experiment result

Reference

Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu,
“ Global Contrast Based Salient Region Detection” ,
IEEE TRAN PATTERN ANALYSIS AND MACHINE INTELLIGENCE,
VOL. 37, NO. 3, p.569-p.582, MARCH 2015