Final project report\_team24

一、目的

將圖片去背並且移到另一張圖上面

二、圖片概念

圖片由前景及背景構成，組成可由以下公式得到:

I = αF + (1-α)B

I:given image F:foreground B:background α:透明度

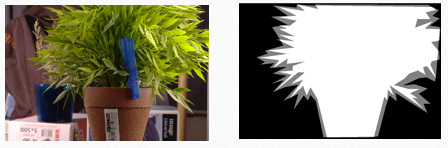
三、實作方法

原圖 -> trimap -> alpha -> 改變背景

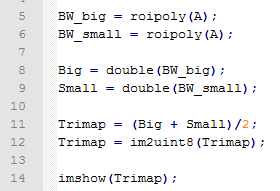
1.原圖->trimap

手動點出內圈及外圈的邊界，最內部為前景；中間為不確定為前景還是背景的模糊地帶；最外面為背景。

Trimap: 三分圖，我們將最前景設為1(白色)；模糊地帶設為0.5(灰色)；背景設為0(黑色)。

Ex: 左邊為原圖；右邊為trimap

程式部分:



A是input image，利用roipoly函數產生可以選取邊界的圖，roipoly選到的部分會將內部設為1、外部設為0，所以將(外圈+內圈)/2就可以得到值為0、0.5、1的三個範圍。

2.get alpha

Learning Based Matting

令Ω = {1, · · · , n}, n is the total number of pixels

labeled pixels Ωl ⊂ Ω for which we know the α values

Ωu = Ω− Ωl

令Ni = {τ1, ・ ・ ・ , τm}.

αi = [ατ1 , ・ ・ ・ , ατj , ・ ・ ・ ατm] T  ，τj ∈ Ni

**f**i =[fiτ1 , ・ ・ ・ , fiτj , ・ ・ ・ , fiτm]T 表示 linear combination coefficients

對於所有i ∈ Ω，假設其alpha matte value αi 可以利用{αj } ( j∈Ni, Ni⊂ Ω 為 α 的neighboring pixels )的線性組合預估

* αi = **f**i Tαi ・ ・ ・ (1)

令α = [α1, · · · , αn]T

ξi =[fi1, · · · , fin]T

* αi = **ξ**i T **α** ・ ・ ・ (2)

令F = [ξ1, · · · , ξn], {**ξ**i}i∈Ω

Eq3 => α = FT α ・ ・ ・ (3)

若F為已知，則得到以下式子的最小quadratic cost即可計算α

arg minα ||α−FTα||2 + c||αl − αl\*|| ・ ・ ・ (4)

其中αl\*代表αl中的已知值

set c = ∞, 使 labeled foreground pixels 為1，background 為0

令C為 n

×n 的 matrix，若j ∈ Ωl 則將其jth diagonal element 設為c，而其他的 diagonal element設為0

α\*為長度為n的vector，其jth element與已知的alpha value j相等，其中j ∈ Ωl

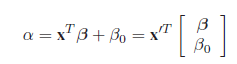
I(n)為n×n的identity matrix

Eq4 => ・ ・ ・ (5)

對Eq5的α作微分，令其等於0，求極小值

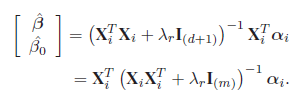
* α =((I(n)− F)(I(n)− F)T + C)−1Cα∗ ・ ・ ・ (6)

令xi 為第i個pixel的feature、 {xj}j∈Ni  x’= [xT 1]T

令・ ・ ・ (7)

其中β= [β1, . . . , βd]T和*β*0 為model coefficients、

Xi = [ xτ1’· · · xτm’]T

利用 ridge regression technique 解 quadratic optimization problem得到β和β0的最佳解・ ・ ・ (8)

其中λr為一個很小的參數

利用Eq(7)、(8)可以得到Eq(2)的fi為

fi =(XiXiT+λrI(m))−1 Xixi’・ ・ ・ (9)

所以現在可以利用fi得到F，進而得到α

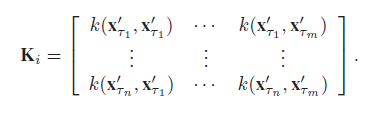
而Eq(7)可以extend到nonlinear：

α = Φ(x)Tβ + β0 ・ ・ ・ (10)

其中β = [β1, . . . , βp]T 、 Φ(x) = [φ1(x), · · · , φp(x)]T

並將Eq(9)的xi和xj的內積替換成k(xi, xj)

令ki = [k(x’τ1 , x’i), · · · , k(x’τm, x’i)]T ・ ・ ・ (11)

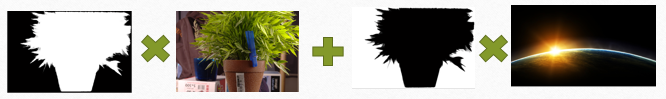
・ ・ ・ (12)

所以fi可以表示成

fi = (Ki + λrI(m))−1ki

3.改變背景

New picture = α×前景 + (1-α)×背景

 = 

背景部分只會處理與前景圖大小相同的地方，其他部分保留原本的背景，而前景圖要放在背景的哪個地方可以手動選擇。

程式部分:

(1)選前景背景

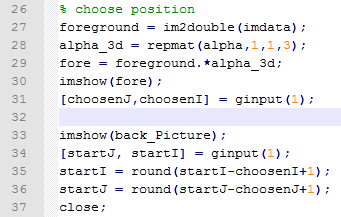


這部分可選擇背景



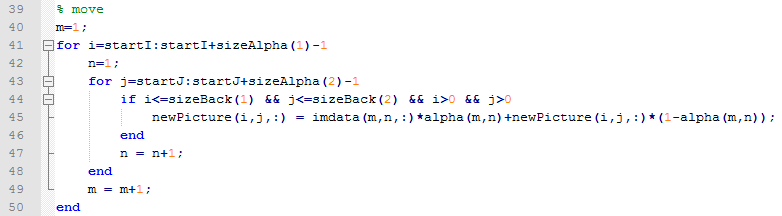
這部分可改變前景

(2)選前景呈現位置



利用ginput可以選擇一個需要的點，在前景上選一個點A(choosenI,choosneJ)，在背景上選一個點B(startI,startJ)，看點A與前景的左上角距離多遠，將點B減掉這個距離後得到點C，在背景上從點C(新的startI,startJ)開始繪製前景，即可選定我們要將前景呈現在哪個地方。

3)將前景移到新的背景上



套用New picture = α×前景 + (1-α)×背景

背景處理與前景一樣大的部分

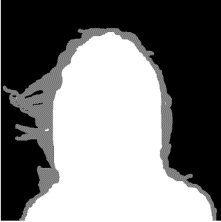
前景放置的位置如果有部分超出背景圖的範圍的話那部份就不會做運算

四、比較

Case1:由原圖產生trimap再丟進algorithm做運算得到α

Case2:由原圖產生scribe再丟進algorithm做運算得到α

(最終實作的是case1)

Ex:由左到右分別是原圖、trimap、scribe

因為一開始我們兩種方法都有使用，後來發現case1的較容易成功，因此就我們做的情況做比較。

產生方面:

Case1:要利用原圖產生trimap感覺比較難，後來找到方法實作，但中間模糊地帶的值根據我看到的資料應該是在0~1之間，所以我自己實作就設為0.5，而且因為要手動畫邊界，較花時間

Case2:產生scribe只要在原圖的前景畫上白線與背景畫上黑線即可，感覺較容易，也比較快

成功率方面:

Case1:只要邊界不要亂畫，幾乎都能成功，差別只在於邊界畫的好，成果會更精準

Case2:對於線條的要求較多，例如用小畫家的話只能用鉛筆，而且對於顏色、圖案太複雜的圖片較難判斷，容易出錯，但由於我們後來專注在case1上，所以對於case2演算法的使用不夠了解可能也是造成錯誤的原因

五、如何合作

我們先各自對上面比較的case1、case2做實作，等選定用哪個之後再一起做，之後有想到什麼功能就一起想怎麼實作，接著一起實作。

做完之後各自根據自己的喜好上網找圖片來做範例，並將2人的成果圖放進報告投影片裡。

報告的部分一個人報功能介紹與實作部分；另一個人報demo部分。

六、參考資料

Learning Based Matting & Source Code: <http://goo.gl/7J4Wsc>