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:

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

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:

```
5   BW_big = roipoly(A);
6   BW_small = roipoly(A);
7
8   Big = double(BW_big);
9   Small = double(BW_small);
10
11  Trimap = (Big + Small)/2;
12  Trimap = im2uint8(Trimap);
13
14  imshow(Trimap);
```

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$\Omega = \{1, \cdots, n\}$, n is the total number of pixels
labeled pixels $\Omega_l \subset \Omega$ for which we know the α values
$\Omega_u = \Omega - \Omega_l$

令$N_i = \{\tau_1, \cdots, \tau_m\}$.
$\alpha_i = [\alpha_{\tau 1}, \cdots, \alpha_{\tau j}, \cdots \alpha_{\tau m}]^T$ ，$\tau j \in Ni$
$\mathbf{f}_i = [f_{i\tau 1}, \cdots, f_{i\tau j}, \cdots, f_{i\tau m}]^T$ 表示 linear combination coefficients
對於所有$i \in \Omega$，假設其alpha matte value α_i 可以利用$\{\alpha_j\}$ ($j \in N_i$, $N_i \subset \Omega$ 為 α 的neighboring pixels)的線性組合預估
⇨ $\alpha_i = \mathbf{f}_i^T \alpha_i$ ・・・(1)

令$\alpha = [\alpha_1, \cdots, \alpha_n]^T$
$\xi_i = [f_{i1}, \cdots, f_{in}]^T$
⇨ $\alpha_i = \boldsymbol{\xi}_i^T \boldsymbol{\alpha}$ ・・・(2)

令$F = [\xi_1, \cdots, \xi_n]$, $\{\boldsymbol{\xi}_i\} i \in \Omega$
Eq3 => $\alpha = F^T \alpha$ ・・・(3)

若F為已知，則得到以下式子的最小quadratic cost即可計算α
$\arg\min_\alpha ||\alpha - F^T\alpha||^2 + c||\alpha_l - \alpha_l^*||$ ・・・(4)
其中α_l^*代表α_l中的已知值
set $c = \infty$, 使 labeled foreground pixels 為1，background 為0

令C為 n×n 的 matrix，若$j \in \Omega_l$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 $j \in \Omega_l$
$I_{(n)}$為 n×n 的 identity matrix

Eq4 => $\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \boldsymbol{\alpha}^T(\mathbf{I}_{(n)}-\mathbf{F})(\mathbf{I}_{(n)}-\mathbf{F})^T\boldsymbol{\alpha}+(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^T\mathbf{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)$ ・・・(5)

對 Eq5 的α作微分，令其等於 0，求極小值
⇨ $\alpha = ((I_{(n)} - F)(I_{(n)} - F)^T + C)^{-1} C\alpha^*$ ・・・(6)

令 x_i 為第 i 個 pixel 的 feature、 $\{x_j\}j\in N_i$ $x' = [x^T\ 1]^T$

令 $\alpha = \mathbf{x}^T\beta + \beta_0 = \mathbf{x}'^T \begin{bmatrix} \beta \\ \beta_0 \end{bmatrix}$ ・ ・ ・ (7)

其中 $\beta = [\beta_1, \ldots, \beta_d]^T$ 和 $\beta 0$ 為 model coefficients、

$X_i = [\ x_{\tau 1}' \cdots x_{\tau m}'\ ]^T$

利用 ridge regression technique 解 quadratic optimization problem

$$\begin{bmatrix} \hat{\beta} \\ \hat{\beta}_0 \end{bmatrix} = \left(\mathbf{X}_i^T\mathbf{X}_i + \lambda_r\mathbf{I}_{(d+1)}\right)^{-1}\mathbf{X}_i^T\boldsymbol{\alpha}_i$$
$$= \mathbf{X}_i^T\left(\mathbf{X}_i\mathbf{X}_i^T + \lambda_r\mathbf{I}_{(m)}\right)^{-1}\boldsymbol{\alpha}_i.$$

得到 β 和 $\beta 0$ 的最佳解 ・ ・ ・ (8)

其中 λ_r 為一個很小的參數

利用 Eq(7)、(8)可以得到 Eq(2)的 f_i 為

$f_i = (X_iX_i^T + \lambda_r I_{(m)})^{-1}\ X_ix_i'$ ・ ・ ・ (9)

所以現在可以利用 f_i 得到F，進而得到 α

而Eq(7)可以extend到nonlinear：

$\alpha = \Phi(x)^T\beta + \beta_0$ ・ ・ ・ (10)

其中 $\beta = [\beta 1, \ldots, \beta_p]^T$ 、 $\Phi(x) = [\varphi_1(x), \cdots, \varphi_p(x)]^T$

並將Eq(9)的 x_i 和 x_j 的內積替換成 $k(x_i, x_j)$

令 $k_i = [k(x'_{\tau 1}, x'_i), \cdots, k(x'_{\tau m}, x'_i)]^T$ ・ ・ ・ (11)

$$\mathbf{K}_i = \begin{bmatrix} k(\mathbf{x}'_{\tau_1}, \mathbf{x}'_{\tau_1}) & \cdots & k(\mathbf{x}'_{\tau_1}, \mathbf{x}'_{\tau_m}) \\ \vdots & \vdots & \vdots \\ k(\mathbf{x}'_{\tau_n}, \mathbf{x}'_{\tau_1}) & \cdots & k(\mathbf{x}'_{\tau_n}, \mathbf{x}'_{\tau_m}) \end{bmatrix}.$$

・ ・ ・ (12)

所以 f_i 可以表示成

$f_i = (K_i + \lambda_r I_{(m)})^{-1}k_i$

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

```
% background picture
back_Picture_name = '.\data\background\test_background.jpg';
```

這部分可選擇背景

```
% foreground picture
fn_im='.\data\foreground\plant.png';
```

這部分可改變前景

(2)選前景呈現位置

```
% choose position
foreground = im2double(imdata);
alpha_3d = repmat(alpha,1,1,3);
fore = foreground.*alpha_3d;
imshow(fore);
[choosenJ,choosenI] = ginput(1);

imshow(back_Picture);
[startJ, startI] = ginput(1);
startI = round(startI-choosenI+1);
startJ = round(startJ-choosenJ+1);
close;
```

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

```
% move
m=1;
for i=startI:startI+sizeAlpha(1)-1
    n=1;
    for j=startJ:startJ+sizeAlpha(2)-1
        if i<=sizeBack(1) && j<=sizeBack(2) && i>0 && j>0
            newPicture(i,j,:) = imdata(m,n,:)*alpha(m,n)+newPicture(i,j,:)*(1-alpha(m,n));
        end
        n = n+1;
    end
    m = m+1;
end
```

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