VFX Final Project
Coordinate for Instant Image Cloning with Foreground Extraction

b02901004 王士元
b02901051 呂弈臻
b02901101 林士庭

1 Introduction

在gradient domain上進行seamless cloning是一個相當廣爲應用的方式，以[3]爲其代表，但這類的方法往往需要透過解一個龐大的poisson equation才能完成，速度是他的主要瓶頸。

在project中，我們參考了[1]的方法來建構一套可以根據使用者在source image GUI上畫出特定區域，並且快速在target image完成image cloning功能的系統，同時允許使用者利用拖曳的方式隨時更改位置，並且任意縮放、旋轉。

考慮到一般情形下，使用者會期望將source image的foreground region標記出來進行image cloning，我們的系統也會採用[2]的方式，擷取出image中foreground的區域，同時提供互動式的GUI讓使用者可以根據系統產生的結果給予進一步的修正。 簡而言之，我們所完成的系統包含這些功能：

- 讓使用者透過GUI在source image圈選出ROI(region of interest)
- 利用grabcut切割出foreground region
- 利用mean-value-coordinate完成快速image cloning
- 提供poisson cloning供比較使用
- 應用於object removal

2 Interactive foreground extraction with GrabCut[2]

2.1 Motivation

本paper根據過去已存在的graph cut應用於forefround extraction的方法，作出改進以提供interactive minimization和接受使用者輸入incomplete labelling，這兩項改進除了可以減少使用者負擔，也可以增進accuracy。

2.2 Problem description

給定一initial trimap T，image array $\mathbf{z} = (z_1, z_2, z_3, \cdots, z_N)$，foreground extraction可以表示爲image segmantation的問題，並且對每個pixel，以 $\underline{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_N)$ 代表每個pixel的opacity value，在此處，我們考慮的是hard segmentation的狀況，也就是$\alpha_n \in \{0,\ 1\}$，0是background，1是foreground。Parameter set $\underline{\theta}$描述foreground以及background pixels的distribution。另外，定義energy function $\mathbf{E}$，兼顧pixel屬於何種label的機率以及鄰近區域內的一致性，故其最小值可反映一個好的segmentation。

因此，foreground extraction的問題可以改寫爲對於energy function的minimization problem。

$$\hat{\underline{\alpha}} = \arg\min_{\underline{\alpha}} \mathbf{E}(\underline{\alpha}, \underline{\theta}) \tag{1}$$

此問題可以透過minimum cut algorithm完成hard segmentation。

2.3 Data modeling

對於整張image $\mathbf{z}$，我們會分別對於foreground以及background region各自建立GMM (Gaussian mixture model)，實作上採用full-covariance K-component GMM ($K = 5$)，其對應的parameter set爲：

$$\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = 0, 1, k = 1, 2, \cdots, K\} \tag{2}$$

where $\pi(\cdot)$ are mixture weighting coefficient. 並且紀錄$\mathbf{k} = \{k_1, k_2, \cdots, k_N\}, k_n \in \{1, \cdots, K\}$以代表每個image pixel分配給哪一個GMM component，並根據α_n決定爲foreground或是background GMM models。

此處所使用的energy function爲Gibbs energy，其包含兩部分，date term U以及smoothness term V:

$$\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}) \tag{3}$$

smoothness term定義爲：

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in\mathbf{C}} \frac{1}{dis(m,n)} \mathbb{1}_{\alpha_m \neq \alpha_n} e^{-\beta\|z_m - z_n\|^2} \tag{4}$$

where $\mathbb{1}_{[\cdot]}$ is the indicator function, $\mathbf{C}$ is the set of neighboring pixels.

式.4會鼓勵具有相似pixel的區域達成一定程度的同調性，另外constant

$$\beta = (2\langle(z_m - z_n)^2\rangle)^{-1} \tag{5}$$

where $\langle\cdot\rangle$ denotes expectation over an image sample. 而data term則定義爲：

$$\begin{aligned} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) &= \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n) \\ D(\alpha_n, k_n, \underline{\theta}, z_n) &= -\log p(z_n|\alpha_n, k_n, \underline{\theta}) - \log \pi(\alpha_n, k_n) \end{aligned} \tag{6}$$

where $p(\cdot)$ is a Gaussian distribution.

2.4 Segmentation algorithm

Initialization

- (Incomplete labelling) Initial trimap from user selection with T_B only, which means $T_F = \emptyset$ and $T_U = \bar{T_B}$.
- Initialize $\alpha_n = 1$ if $n \in T_U$ and $\alpha_n = 0$ if $n \in T_B$.
- Initialize foreground and background GMM from sets $\alpha_n = 1$ and $\alpha_n = 0$ respectively.

Iterative minimization

1. Assign GMM components to each pixels:

$$k_n := \arg\min_{k_n} D_n(\alpha_n, k_n, \underline{\theta}, z_n), \forall n \in T_U \tag{7}$$

2. Learn GMM parameters from data $\mathbf{z}$:

$$\underline{\theta} := \arg\min_{\underline{\theta}} U(\underline{\theta}, \mathbf{k}, \underline{\theta}, \mathbf{z}) \tag{8}$$

3. Estimate segmentation:

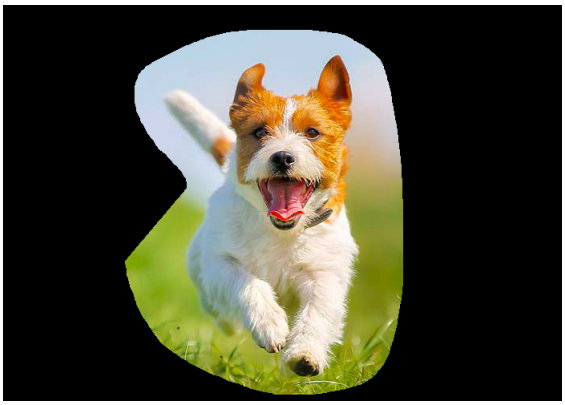
$$\min_{\{\alpha_n : n \in T_U\}} \min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) \tag{9}$$

4. Begin from step 1, until convergence.

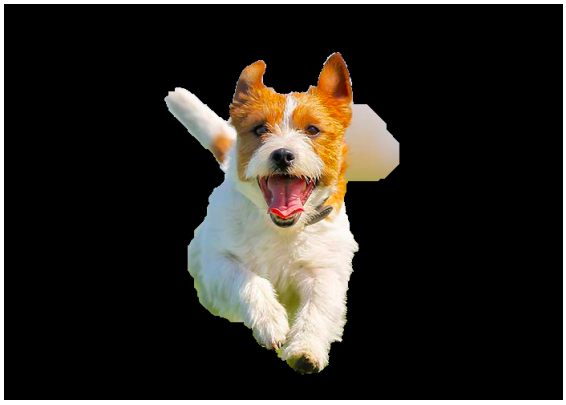
User editing Mark foreground and background by user, then perform entire iterative minimization.

2.5 Results

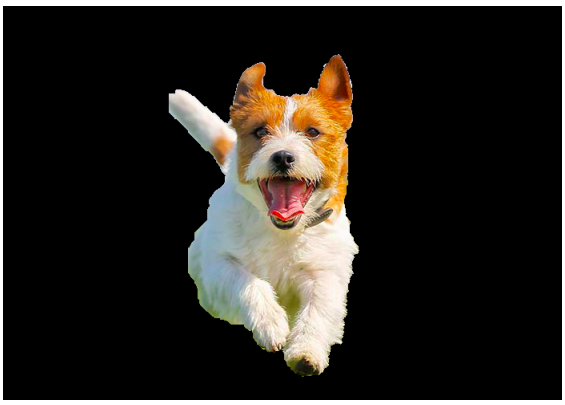
(a) Original image



(b) Lasso provided by user



(c) outcome of first iteration



(d) refined outcome

3 Poisson cloning[3]

原作者提出此方法的理念在於，使用Laplacian operator所取出的2階變化量對於人的感知具有相當的影響力，並且疊加在原影像上並不會產生過於突兀的變化；另一方面，當給定一個區域其Laplacian以及boundary上的scalar function，此時內部區域根據Poisson's equation可以得到唯一的解。

3.1 Problem description

如Figure 2考慮一張 target image S 與其 pixel values function f^*，以及 source image G 與其 pixel values function g ，使用者選定的待處理 cloning 區域爲 Ω，對於每一個 pixel p，定義 N_p 爲 p 四個方向上相鄰的 pixel。可以將此一關係表示爲: pair (p, q) such that $q \in N_p$。而Ω的boundary則定義爲 $\partial\Omega = \{p \in S \backslash \Omega | N_p \cap \Omega \neq \emptyset\}$。我們的目標是要找出$\Omega$區域的pixel value，即$f\ |_\Omega = \{f_p,\ p \in \Omega\}$。

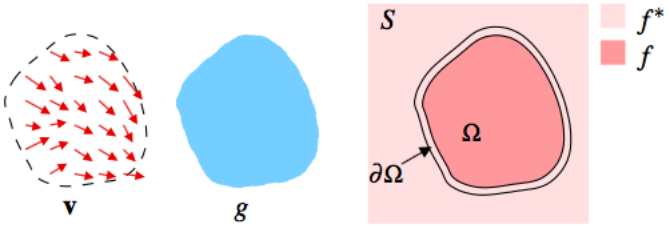


Figure 2: For image S, f is an unknown function on Ω, and f^* is a known function. $\mathbf{v}$ is a guidance field on Ω, which might be gradient of g.

爲了使最後結果中的Ω區域能與 target image 完美融合，基於人類感知系統對於像素的二階變化量較爲敏感此一假設，以及 boundary points 上之像素值需與 target image 完全相同此一限制，$f\,|_\Omega$ 必須符合下式之 Poisson equation，

$$\Delta f = \mathrm{div}\nabla g,\ f\,|_{\partial\Omega} = f^* \tag{10}$$

對於此Dirichlet boundary problem，我們可以將式.10 改寫爲一離散的optimization problem：

$$\min_{f|_\Omega} \sum_{(p,q)\cap\Omega\neq\emptyset} (f_p - f_q - v_{pq})^2,\ \text{with}\ f_p = f_p^*, \forall p \in \partial\Omega \tag{11}$$

此處v_{pq}可以視爲是guidance field $\mathbf{v}(\frac{p+q}{2})$在$[p,q]$方向上的projection，也就是$v_{pq} = \mathbf{v}(\frac{p+q}{2})\cdot\overrightarrow{pq}$。並且，他的解將會滿足下列linear equation。

$$\forall p \in \Omega,\ |N_p| f_p - \sum_{q\in N_p\cap\Omega} f_q = \sum_{q\in N_p\cap\partial\Omega} f_q^* + \sum_{q\in N_p} v_{pq} \tag{12}$$

此處的$|N_p|$代表的意義是pixel p的周遭有多少有效的pixel，也就是尚未超出影像邊界區域的pixel數目。

3.2 Implementation

在實作上，我們會將g視作target image的pixel value，可以得到：

$$\begin{aligned} \mathbf{v} &= \nabla g \\ v_{pq} &= g_p - g_q,\ \forall (p,q) \end{aligned} \tag{13}$$

進而將之整理成下列演算法:

Algorithm 1 Poisson cloning

Input: src_img: source image
Input: tar_img: target image
Output: Interpolateded image

procedure PoissonCloning(src_img, tar_img)
  Compute and locate blending region Ω
  for each $p \in \Omega$ **do**
    $|N_p| \leftarrow 0,\ v_{pq} \leftarrow 0,\ f_p \leftarrow tar\_img[p],\ g_p \leftarrow src\_img[p]$
    for each $q \in N_p$ **do**
      if q is a valid pixel **then** ▷ Valid means pixel has meaningful value.
        $|N_p| += 1,\ f_q \leftarrow tar\_img[q],\ g_q \leftarrow src\_img[q],\ f_q^* = 0$
        if q lies in Ω **then**
          $v_{pq} += g_p - g_q$ ▷ Guidance field is gradient image.
        else if q lies in $\partial\Omega$ **then** ▷ g_q not a valid pixel.
          $v_{pq} += 0$
          $f_q^* \leftarrow f_q$
        end if
      end if
    end for
    Construct linear equation of pixel p.
  end for
  for each channel **do**
    $x \leftarrow$ Solve linear equation.
  end for
  $f \leftarrow x$
end procedure

4 Poisson cloning with MVC[1]

4.1 Motivation

以 poisson cloning 進行 seamless cloning 可以達到相當不錯的結果，然而因爲其在實作中需要解一個相當龐大的 linear equation 來得到 poisson equation 的解，因此其運算也就非常耗時。在[1]這篇論文中，作者觀察到要解式.10，即等價於解以下之 laplace equation，

$$\begin{aligned} \Delta\widetilde{f} = 0,\ \widetilde{f}\ |_{\partial\Omega} &= f^* - g \\ f\ |_{\Omega}\ &= g + \widetilde{f} \end{aligned} \tag{14}$$

而此一方程式其實表示 poisson cloning 是在建立一個均值函數 $\widetilde{f}$ 來將 target image 與 source image 在 $\partial\Omega$ 上的像數值差異 interpolate 到整個 blending 區域上。而 mean-value coordinates 則是這類 boundary interpolate problem 之 均值函數解的一個相當好的近似。

4.2 Implementation

考慮一個封閉的多邊形 boundary $\partial P = (\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_m = \mathbf{p}_0), \mathbf{p}_i \in \mathbb{R}^2$，一點 $\mathbf{x} \in \mathbb{R}^2$

對 $\partial\Omega$ 之 mean-value coordinates $\lambda(\mathbf{x}) \in \mathbb{R}^m$ 可以下式表示，

$$\begin{aligned} \lambda_i(\mathbf{x}) &= \frac{w_i}{\sum_{j=0}^{m-1} w_j} \\ \text{where } w_i &= \frac{\tan(\alpha_{i-1}/2) + \tan(\alpha_i/2))}{\|\mathbf{p_i} - \mathbf{x}\|} \end{aligned} \tag{15}$$

其中 α_j 爲 $\mathbf{p_i}$, $\mathbf{x}$ 與 $\mathbf{p_{i+1}}$ 在二維空間中的夾角。而式.14 中 laplace equation 的 均值函數解 $\widetilde{f}$ 其在各點 $\mathbf{x} \in \Omega$ 上的取值即可在已知 $(f^* - g)\ |_{\partial\Omega}$ 的情況下以 MVC 爲權重 interpolate 來得到，如下式：

$$\widetilde{f}(\mathbf{x}) = \sum_{i=0}^{m-1} \lambda_i(\mathbf{x})(f^* - g)(\mathbf{p}_i) \tag{16}$$

其中 $(\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_m = \mathbf{p}_0) = \partial\Omega,\ \mathbf{p}_i \in \mathbb{R}^2$。而因爲通常在cloning 時 boundary 在使用者決定過後就不會再改變了，因此我們可以將計算 MVC 視爲 image 的 preprocessing，在整個演算法只需計算一次，進而可以大大提升演算法的效率。

4.3 Optimization

4.3.1 Adaptive Mesh

藉由 mean-value coordinates，我們已經可以將 cloning 的運算複雜度降爲 O(nm)，其中 n = $|\Omega|$，m = $|\partial\Omega|$。不過，因爲以 MVC 計算出的數值幾乎非常平滑，我們可以另外藉由對 Ω 建立 triangular mesh，並只對在 mesh 頂點上的少數座標點計算 MVC，而其他並非 mesh 頂點的座標點之數值則可利用其對應到的三角形的三個頂點以 barycentric coordinates內差得到。此一方式可進一步爲演算法加速至 O(m^2)。而在這邊我們使用了 python 的 toolkit triangle 與 scipy 來爲 Ω 建立 trianglular mesh 與計算 barycentric coordinates。

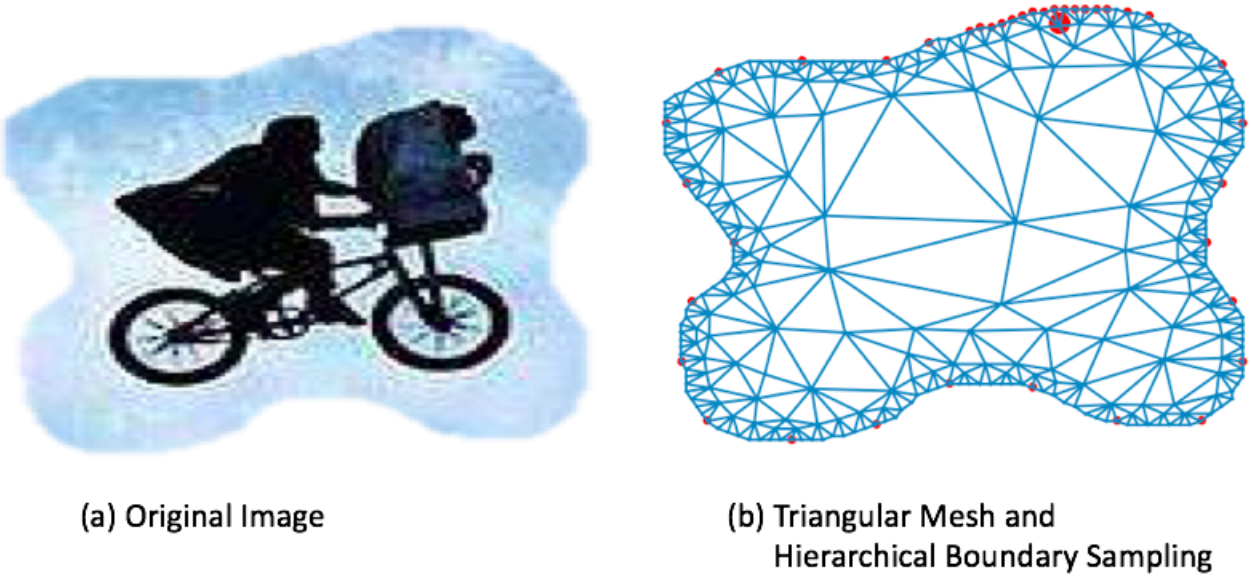
4.3.2 Hierarchical Boundary Sampling

由式.15，我們可以看出 MVC 會隨著與邊界點的距離增加而快速遞減，因此每個 blending 區域內的點在做 interpolate 時並不需要參考所有的邊界點，而可以以與距離成反比的密度 sample 邊界點，並只與 sample 出的邊界點作 interpolate，如此一來即可以再度減少所需的計算量。

而實作的方式我們則是參考[1]。在對某一點 $\mathbf{x} \in \Omega$ 尋找sampling boundary 時，對於整個邊界點的集合 $\partial\Omega$，我們首先會以均勻地以 s_1 的精度 sample 邊界點，建立包含最少邊界點的第一階層 L_1，其後，對每階層的每一點我們都會以下述方式判斷在此點附近需不需要更進行更精細的 interpolate。 假設在 L_i 中有一邊界點 $\mathbf{p_j^k}$，而其在 L_i 中前後分別爲 $\mathbf{p^k_{j-s_i}}$ 與 $\mathbf{p^k_{j+s_i}}$，若邊界點不符合下列任一式，

$$\begin{aligned} \|\mathbf{p}_i^k - \mathbf{x}\| &< \epsilon_{\mathbf{dist}} \\ \angle\mathbf{p}^k_{j-s_i}, \mathbf{x}, \mathbf{p}^k_j &< \epsilon_{\mathbf{angle}} \\ \angle\mathbf{p}^k_j, \mathbf{x}, \mathbf{p}^k_{j+s_i} &< \epsilon_{\mathbf{angle}} \end{aligned} \tag{17}$$

我們即會將$\mathbf{p^k_{j-s_i/2}}$ 與 $\mathbf{p^k_{j+s_i/2}}$ 加入下一階層，也就是說，在$\mathbf{p_j^k}$ 附近的 interpolate 會更加精細。



5 Our results

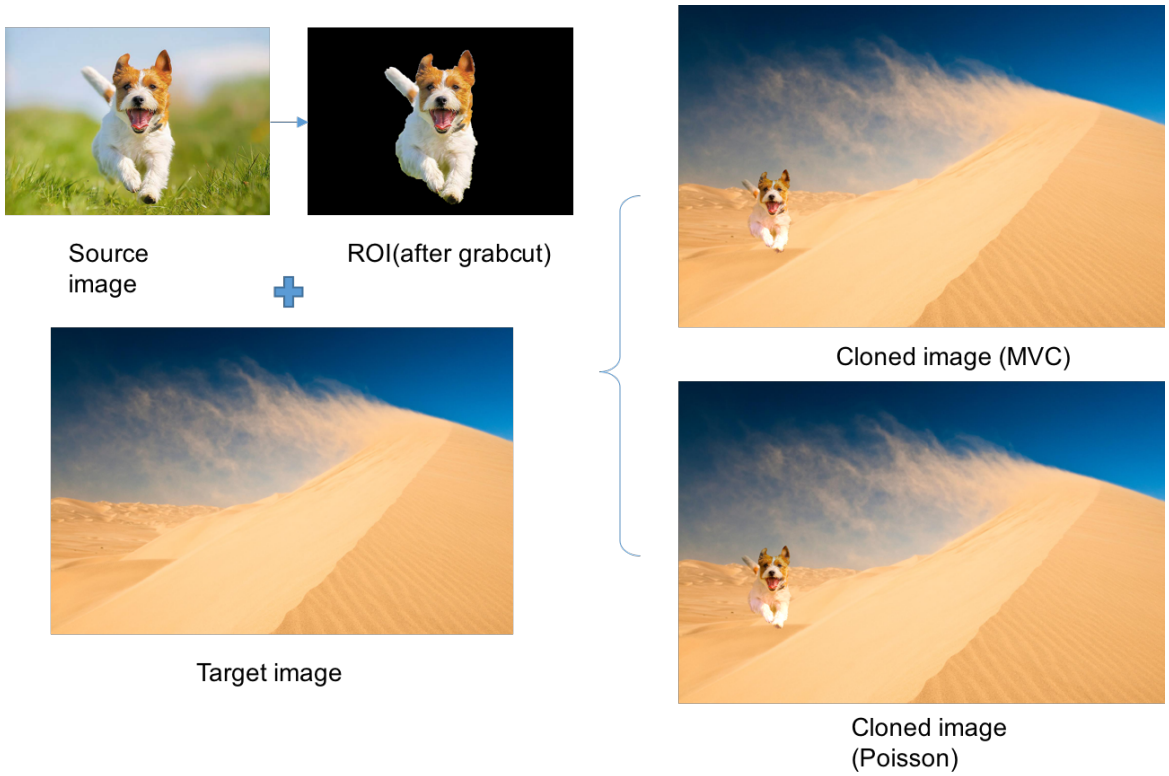


Figure 3: Cloning process, including MVC and Poisson cloning

References

[1] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, D. Lischinski, *Coordinates for Instant Image Cloning*, ACM SIGGAPH 2009.

[2] Carsten Rother, Vladimir Kolmogorov, Andrew Blake, "*GrabCut*" — *Interactive Foreground Extraction using Iterated Graph Cuts*, ACM SIGGRAPH 2004.

[3] Patrick Perez, Michel Gangnet, Andrew Blake, *Poisson Image Editing*, Microsoft Research UK.