R94944001 謝永桓

Report of Machine Learning for multimedia

Group Topic: Graph Models & Belief Propagation

Two Practical Example of BP

Paper

[1]

Topic :	Efficient Belief Propagation for Early Vision
Author :	Pedro Felzenszwalb, Daniel Huttenlocher
From :	CVPR, 2004

[2]

Topic :	An Iterative Optimization Approach for Unified Image Segmentation and Matting
Author :	Jue Wang, Michael Cohen
From :	ICCV, 2005

這兩篇 Paper 是實用之前討論的 Graph Model，及應用 Belief Propagation 計算 Markov random fields 的 performing inference。兩篇作者提供的想法是分別將不同的問題歸類成符合 Markov Random Field (MRF)的模型，再依要解決的問題而應用 Belief Propagation 方法，反覆的以 Belief (Message)傳遞 update，估計出所需最佳結果。

Efficient Belief Propagation for Early Vision

[1]中主要討論的問題是應用在解決 Early Vision 的問題。Early Vision 包括 stereo, optical flow 以及 image restoration 等視覺問題。其中應用的方法如下：
首先將問題 format 成 MRF 的模型，即在 MRF 模型下計算 Energy Minimization。由於在 MRF 中 Minimization 問題是 NP-hard，因此應用 Belief Propagation 的演算法做估算的算法。我們將問題整理如下：

$$E(f) = \sum_{(p,q)\in N} V(f_p, f_q) + \sum_{p\in \mathrm{P}} D_p(f_p)$$

其中 P 是 set of pixels in image，L 是 set of labels，N 是 the edge in the four-connected image grid graph。f 代表指定的 labeling，$f_p \in L$ 對應於

pixel $p \in P$，表示對某個 pixel P 我們指定的 label f_p。$V(f_p, f_q)$ 對應於 Graph model 中的 discontinuity cost，代表對兩個相鄰的 pixel p 和 q 來說，assign 兩種不同的 label 所需的 cost，。$D_p(f_p)$ 對應於 Graph model 中是 data cost ，是 assign f_p 給 pixel p 所需的 cost。上式 $E(f)$ 代表由 labeling 產生的 energy function，代表 quality 的結果。因此我們的問題是對應這樣定義的 MRF，估計某種 label 可以產生最小的 energy。

Paper 內用 max-product BP algorithm 來計算最小的 cost，我們採用先前的 formulation 來表示 BP 中的 message

$$m_{pq}^{t}(f_q) = \min_{f_p}\left(V(f_p, f_q) + D_p(f_p) + \sum_{s \in N(p)\setminus q} m_{sp}^{t-1}(f_p)\right)$$

其中的 m_{pq}^{t} 代表在時間 t 時由 node p 傳遞到 node q 的 message。$N(p) \setminus q$ 代表 p 附近除了 q 的 node。在 T 個 iteration 後，

$$b_q(f_q) = D_q(f_q) + \sum_{p \in N(q)} m_{pq}^{T}(f_q)$$

最後每個 node 都可算出最小的 $b_q(f_q)$ 來決定所要的 label f_q^* 。

基本上過去的 paper 中有對 early vision 產生的 graph model 做演算法上的分析，其中包括應用 belief propagation。本篇的重點是放在將計算 BP 的方法做加速，將原本共需要 $O(nk^2T)$，包括對 T 次每次需要 $O(n)$ 時間的 iteration，以及每次 compute message 要花 $O(k^2)$。經過加速後，可以有 $O(nk)$ 的時間。

改善的方法有三個部分：

(1). Computing Message :

首先討論的是在相鄰 pixel 間的 cost $V(f_p, f_q)$，對一般的 low level vision problem cost function 基本上是兩個 label f_p 和 f_q 的差值。因此，如果我們把這個 cost 訂成 $V(f_p, f_q) = \begin{cases} 0, f_p = f_q \\ d, otherwise \end{cases}$，則 cost function 可改寫成

$m_{pq}^{t}(f_q) = \min\left(h(f_q), \min_{f_p} h(f_p) + d\right)$，其中

$h(f_p) = D_p(f_p) + \sum m_{sp}^{t-1}(f_p)$，如此計算 message 的時間需要

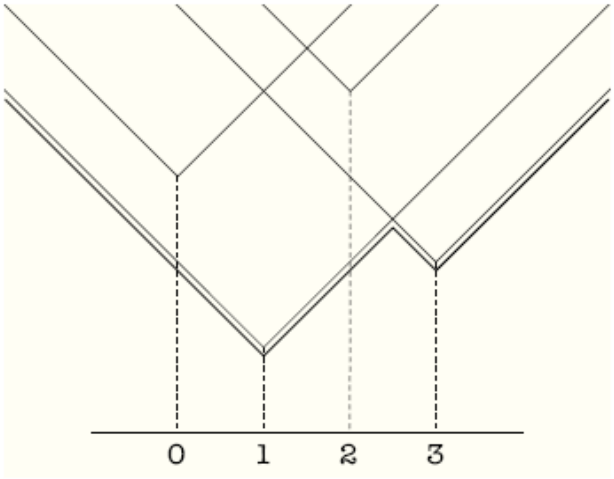
$O(k)$ for k levels。

接著考慮$V(f_p, f_q)$是一個 truncated linear model，即

$V(f_p, f_q) = \min\left(s\left\|f_p - f_q\right\|, d\right)$，這裡的 d 即代表線性增加的 cost 上限。這裏考慮兩種情形，如果相差超過定的 cost 上限，情況就像上面考慮的情形，花的時間因此只需$O(k)$。另外要是

$V(f_p, f_q) = s\left\|f_p - f_q\right\|$，計算 message 的 formulation 就變成

$m_{pq}^t(f_q) = \min_{f_p}(s\left\|f_p - f_q\right\| + h(f_p))$， $f_q - m_{pq}^t(f_q)$的關係圖如下：



其中各條不同的V形線是因爲 f_p 所選的值不同而產生，所以需要計算的是在選不同的 f_q 時對應到 message 的最小值。Paper 內用以下的 Two-way pass algorithm :

for f_q **from** 1 **to** $k-1$:

$$m(f_q) \leftarrow \min(m(f_q), m(f_q - 1) + s).$$ (forward)

for f_q **from** $k-2$ **to** 0 :

$$m(f_q) \leftarrow \min(m(f_q), m(f_q + 1) + s)$$ (backward)

舉例來說，一開始最小的 message 值爲 V 形頂點的值$h(f_p)$，在上圖中 f_q 值從 0-3 可看成(3,1,4,2)，在 forward 的過程後成爲(3,1,2,2)，再經過

backward 後變成(2,2,2,2)，即上圖粗線處。這樣 2-way pass 後可以找出對應到最小 message，q 的 level 值。這裡 2-way 的方法是由於 cost 訂爲 linear 的 formulation，因此只要參考左右的最小值就可決定自己本身的 min，而演算法所需的時間爲 $O(k)$ 。

在這邊的討論是在目前我們考慮的 early vision problem 中，可以用 linear 的方法在 process each message 時將計算時間由 $O(k^2)$ 減少到 $O(k)$ 。

(2). 第二部分是利用 bipartite graph BP 來加速，在這邊將計算時的 grid 分爲兩組(A,B)，而傳遞的 message 就只有兩種: A -> B 和 B -> A，因此在每一次 message 的傳遞(update)時，就只需要更新某一種的 message 就可以。在設計時，就分別記錄兩種不同的 message，再隨 iteration 的過程更新。

$$\bar{m}_{pq}^{t} = \begin{cases} m_{pq}^{t} & \text{if } p \in A \text{ (if } p \in B\text{)} \\ m_{pq}^{t-1} & \text{otherwise} \end{cases}$$

如此大約可以減少一半計算的時間。

(3). Multiscale BP : 這部分是利用減少 message 傳遞的 iteration 技術來加速。採用的做法是 Coarse-to-fine，將計算的範圍由大至小切細成不同 size 的 MB，計算 MB 中 update 的 message，因此就可以解決每個 pixel 都要傳遞 message 的計算量。

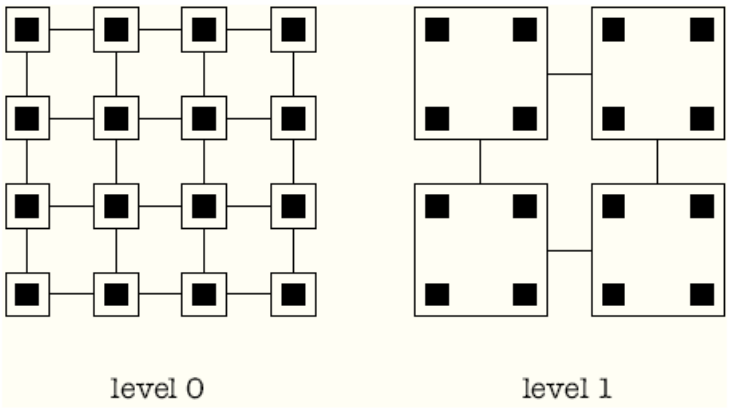


Figure 2: Illustration of two levels in our coarse-to-fine method. Each node in level i corresponds to a block of four nodes in level $i-1$.

在計算由大到小的 message 時，則將大 block 內每個小 block 的 message 指定爲大的 message:

$$r_{p,i-1}^{0} \leftarrow r_{p',i}^{T} \qquad l_{p,i-1}^{0} \leftarrow l_{p',i}^{T}$$

$$u_{p,i-1}^{0} \leftarrow u_{p',i}^{T} \qquad d_{p,i-1}^{0} \leftarrow d_{p',i}^{T}$$

另外在計算時需要的 data cost $D_b(f_b) = \sum_{p \in b} D_p(f_b)$，如此的話則每個切出的 MB 可以有多個造成 low cost 的 level 值，而不會因爲合併計算而和單一 pixel 計算的結果隨 iteration 有不趨近的走向。另外在計算的過程中，會比單一 pixel 的計算 update 的結果收斂至相近的結果。(如圖)

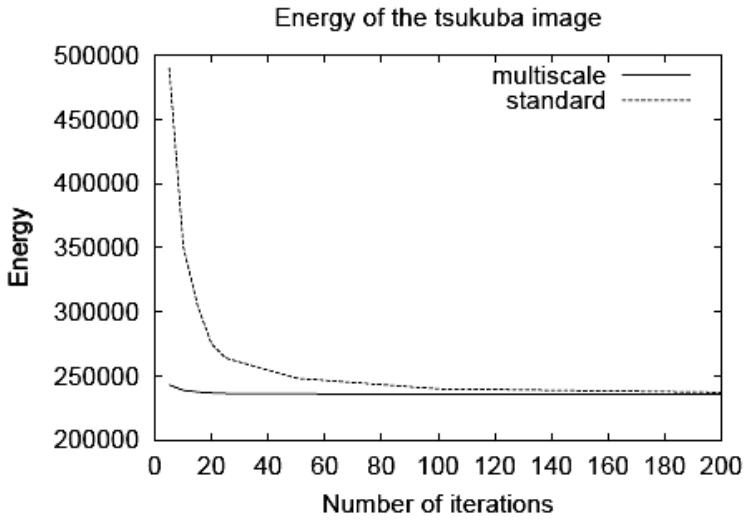


Figure 3: Energy of stereo solution as a function of the number of message update iterations.

實驗結果

下面圖是採用了加速的演算法做幾個 early vision problem: Stereo,Optical flow, Image Restoration，也列出加速後的時間比較：

Stereo:

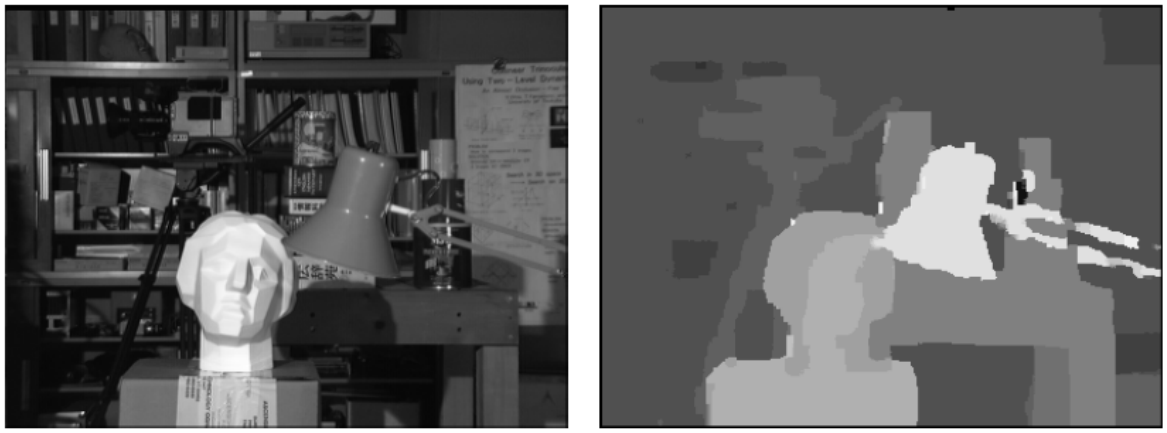


Figure 4: Stereo results for the Tsukuba image pair.

Tsukuba		Sawtooth		Venus		Map	
Rank	Error	Rank	Error	Rank	Error	Rank	Error
8	1.86	7	0.97	4	0.96	9	0.33

Optical flow:

Figure 5: Optical flow results for the Lab image pair.

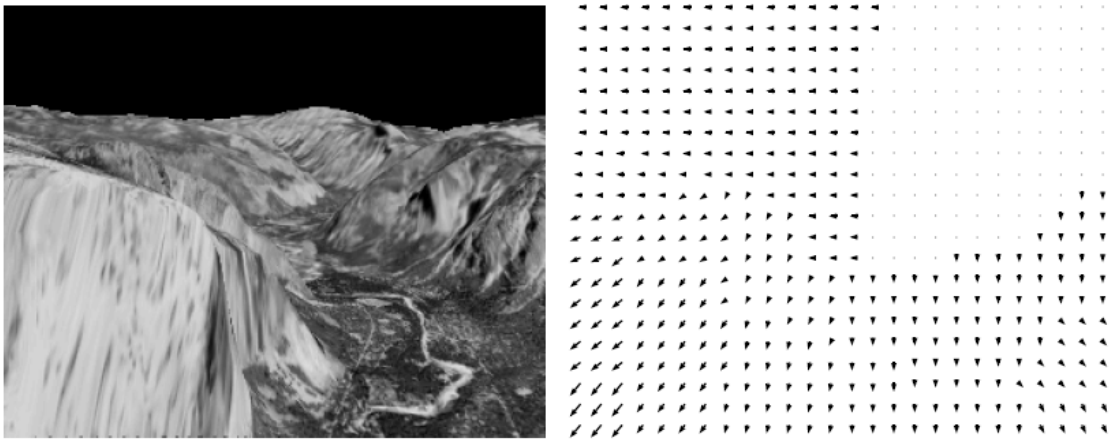
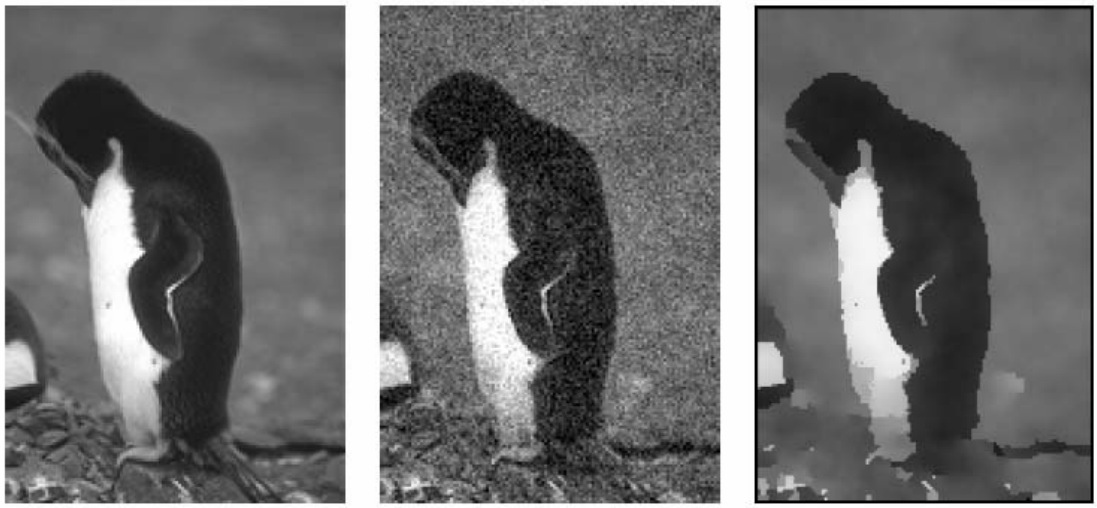


Figure 6: Optical flow results for the Yosemite image pair.

Restoration:



Original Corrupted Restoration

Figure 7: Restoration results for the penguin image.

An Approach for Unified Image Segmentation and Matting

[2]中主要在應用 BP 的做法到另一個 graph 的問題：Segmentation。這篇 Paper 應用 BP 到 Matte Estimation 上。主要是根據以下的式子，

$$I(z) = \alpha_z F(z) + (1 - \alpha_z) B(z)$$

其中 $I(z)$ 指的是真正 pixel 的 color；$F(z)$ 爲 foreground color；$B(z)$ 爲 background color。由於會要求使用者在一開始輸入少數幾個 pixel 做 foreground 和 background 的 color，因此只要算出對每個 pixel 對應的 α 值，即可對 pixel 依 α 值做 matting 的效果。詳細做法如下：

Iterative Belief Propagation for Image Matting

(1). 一開始會需要使用者 mark 出少部分的 pixel color 當作 foreground background color，在此定義 Uc is the pixel set of α estimated，Un is the pixel set of non-estimated。另外對每個 pixel 指定 u[0,1]代表不確定的分數：0 爲確定、1 爲不確定。現在將 user mark 起來的 pixel 放到 Uc，其他放到 Un。

(2). 對每個 marked 的 foreground background color，compute 個別的 GMMs，再 assign 到單一的 Gaussian 以做更進一步的 global sampling.。

(3). 重複下面的步驟，直到 Un 爲空的以及全部的不確定分數 u 爲最小

A. 如果 Un 不是空的，從 Un 中取出和 Uc 裡相鄰的 15 個 pixel 加入 Uc，並將取出的 pixel 轉成 Uc 中計算的 transform

B. 依 Uc 中的 pixel 建立 MRF model，建立的方法如下式：

$$V = \sum_{p \in \varphi} V_d(\alpha_p) + \sum_{p,q \in \varphi} V_s(\alpha_p, \alpha_q)$$

其中 p,q 爲相鄰的 node，V_d 代表 data energy，表示說對 p 而言估計的 α 好壞；V_s 代表 smoothness energy，表示在相鄰的 p 和 q 中間不連續的 α 值變化。在這裡希望的最佳解是在每次的 iteration 是將上面的 total energy V 求最小值。

C. 對 MRF 中每一個 node，依前景和背景的 colors 做 neighborhood sampling 和 global sampling，並計算 data cost。Group sampling 的 data cost 計算如下：

$$V_d(\alpha_p^k) = 1 - \frac{L_k(p)}{\sum_{k=1}^{K} L_k(p)}$$

其中 $L_k(p)$ 指的是根據 alpha levels 的機率模型，和每個對應某個 k 值 α^k 的前景背景的 weight，計算出的 likelihood。計算公式如下：

$$L_k(p) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_i^F w_j^B \cdot \exp\left(-d_c\left(C_p, \alpha^k F_i^p + (1-\alpha^k)B_j^P\right)^2 \Big/ 2\sigma_d^{k\,2}\right)$$

其中 N 代表用來計算 weight 的 samples 個數，F 和 B 是前景背景的 samples，Cp 是 pixel p 真正的 color，用 dc 來計算在 RGB 上色彩的 Euclidian distance。Covariance σ_k^d 則是用前景的 σ_F 和背景的 σ_B `依 $\alpha_k\sigma_F + (1-\alpha_k)\sigma_B$ 計算出來的。最後前景和背景的 weight 是依下式計算：

$$w_i^F = (1-u(p_i)) \cdot \exp\left(\frac{-s(p,p_i)^2}{\sigma_w^2}\right)$$

其中 s(.)代表兩點的 spatial distance，而 $\sigma_w = r/2$ 代表以 pixel p 爲圓心訂出的 neighborhood area 半徑。

另外 neighborhood term 來 sample smooth cost 定義如下式：

$V_s(\alpha_1, \alpha_2) = 1 - \exp\left(-(\alpha_1 - \alpha_2)^2 / \sigma_s^2\right)$ ，這裏 $\sigma_s = 0.2$

D. 在定義了 data cost 和 smooth cost 後，以 MRF 來跑 Max-Product BP algorithm 來計算現在在 Uc set 中的 matte，這裡將 message 定義如下：

$$m_{pq}^t(k_q) = c \cdot \max_{k_p}\left(V_s\left(\alpha_p^{k_p}, \alpha_q^{k_q}\right) \cdot V_d\left(\sigma_p^{k_p}\right) \cdot \Pi_{\tau \in H(p)\setminus q} m_{\tau p}^{t-1}(k_p)\right)$$

其中 c 是 normalization factor，最後得到

$b\left(\alpha_p^{k_p}\right) = cV_d\left(\sigma_p^{k_p}\right) \cdot \sum_{q \in H(p)} m_{qp}^T(k_p)$ 來推算出 α_p^*

E. 最後對每次計算後得到的結果重新推定不確定數 u*，公式如下：

$$u^*(p) = 1 - \left(w_i^{F*} \cdot w_i^{B*}\right)^{\frac{1}{2}}$$

其中前景和背景的 weight 計算如下：

$$F^*(p), B^*(p) = \arg\min_{F_i^p, B_j^p}\left(C_p - \alpha_p^* F_i^p - \left(1-\alpha_p^*\right)B_j^p\right)^2$$

重複下一個 iteration

(4). 結束

實驗結果

下面將 paper 的做法和參考的 Bayesian Matting 做法做比較

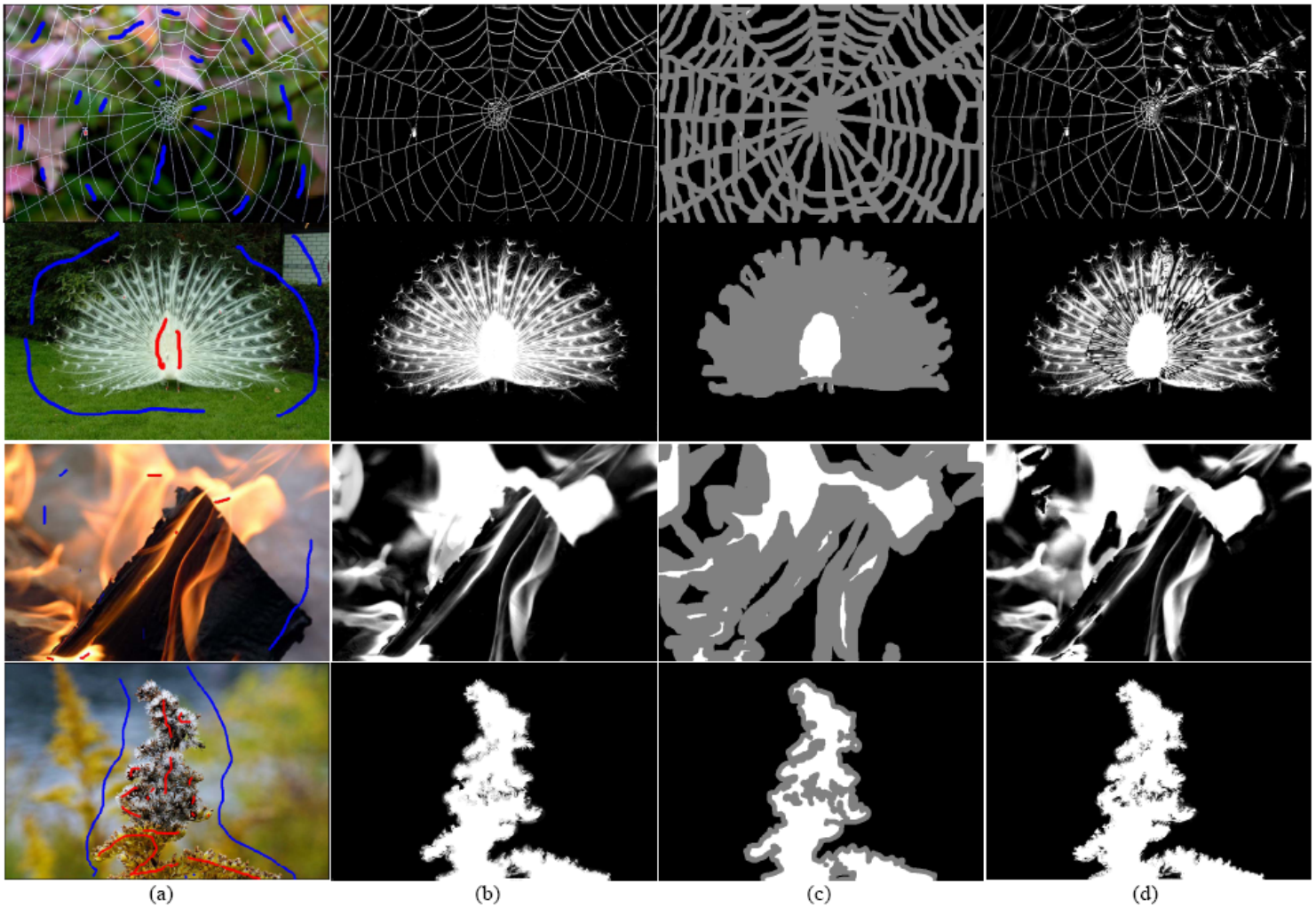


Figure 3. (a). Original images with user specified foreground and background strokes. (b). Extracted alpha mattes by our approach. (c). For Bayesian matting, complex trimaps must be manually created by the user. (d). Bayesian matting results based on trimaps in (c).

下面是背景和前景顏色較近的結果，paper 中的做法可在相同的 trimap 下得到較好的結果

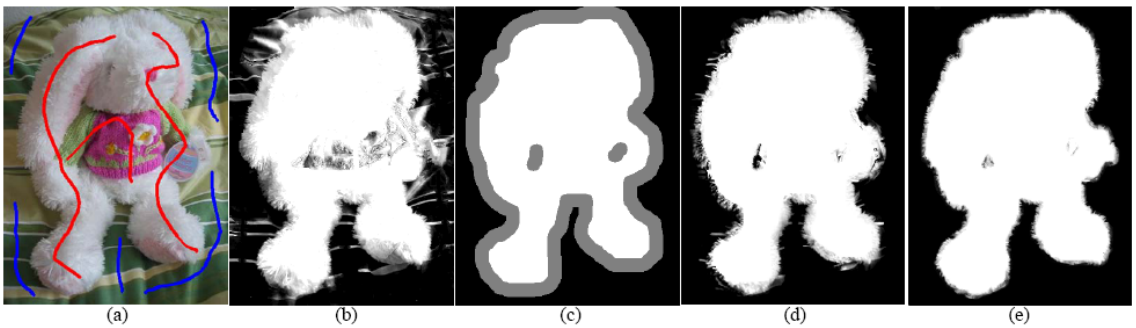


Figure 5. (a). Original image with foreground and background strokes. (b). Extracted matte by our approach is erroneous due to color ambiguity. (c). The user specified rough trimap. (d). Bayesian matting result based on trimap (c). (e) Our matting result based on trimap (c).

最後是將前景切割出來，套上新背景的結果

Figure 4. Composite images with new backgrounds.