Implementation details for Transfusive Image Manipulation

Kaan Yücer[1,2] Alec Jacobson[1] Alexander Hornung[2] Olga Sorkine[1]
[1]ETH Zurich [2]Disney Research, Zurich

Algorithm 1: Content-Aware Bounded Biharmonic Weights

Inputs:
- $I_s \in \mathbb{R}^{w\times h\times 3}$ source LAB image
- $R_s \in \mathbb{R}^{w_t\times h_t}$ ROI image
- m number of weights

Outputs:
- $\mathbf{w_k} \in \mathbb{R}^{w\times h},\ k=1,\dots,m$ weight functions, $m \approx$ #independently moving pieces between source and target

```
// seed selection
G ← gradient_magnitude(I_s) + gradient_magnitude(R_s == 0)
// blur gradient
for t = 1,...,5 do
    G ← G + conv(G, gaussian_kernel(σ ≈ 9))
end
confidence image C ← R_s .* (1 − G)
handles H ← {}
for k = 1,...,m do
    H ← H ∪ arg max_p C(p)
    C ← C − gaussian_kernel(μ = p, σ = sqrt(|R_s > 0|/(πm)))
end
// weight computation
Ω ← triangulate(R_s > 0)
for v ∈ Ω do
    v ← (x, y, sI_s(x,y,1), sI_s(x,y,2), sI_s(x,y,3))      // s ≈ 16
end
// Compute Q
(L, M) = ??(Ω)
Q ← L M^{-1} L
for k = 1,...,m do
    Solve using MOSEK:
```

$$\arg\min_{\mathbf{w}_k} \ \mathbf{w}_k^T Q \mathbf{w}_k$$
$$\text{subject to:}\quad \mathbf{w}_k|_{H_\ell} = \delta_{k\ell} \qquad \ell = 1,\dots,m$$
$$0 \le \mathbf{w}_k(\mathbf{p}) \le 1, \qquad \forall \mathbf{p}\in\Omega$$

```
end
// enforce partition of unity
for k = 1,...,m do
```

$$\mathbf{w}_k(\mathbf{p}) \leftarrow \frac{\mathbf{w}_k(\mathbf{p})}{\sum_{\ell=1}^m \mathbf{w}_\ell(\mathbf{p})} \qquad \forall \mathbf{p}\in\Omega$$

```
end
```

Function computeLM(Ω)

```
// Compute L and M, for given mesh Ω
for Triangle (i, j, k) in Ω do
    l_ij ← ||v_i − v_j||,  l_jk ← ||v_j − v_k||,  l_ki ← ||v_k − v_i||
    r ← 1/2 (l_ij + l_jk + l_ki)                          // semi-perimeter
    A_ijk ← sqrt(r(r − l_ij)(r − l_jk)(r − l_ki))         // area
    cot_ij ← 1/4 (l_jk^2 + l_ki^2 − l_ij^2) / A_ijk
    cot_jk ← 1/4 (l_ki^2 + l_ij^2 − l_ki^2) / A_ijk
    cot_ki ← 1/4 (l_ij^2 + l_jk^2 − l_ki^2) / A_ijk
    for each of 6 permutations (a, b, c) of (i, j, k) do
        L(a, b) ← L(a, b) − 0.5 * cot_ab
        L(a, a) ← L(a, a) + 0.5 * cot_ab
        if cot_ij ≥ 0 and cot_jk ≥ 0 and cot_ki ≥ 0 then
            M(a, a) ← M(a, a) + 1/8 l_ab^2 cot_ab
        else
            M(a, a) ← M(a, a) + { 1/8 A_ijk  if cot_bc ≥ 0
                                  1/4 A_ijk  otherwise }
        end
    end
end
return (L, M)
```

Algorithm 2: Initialization

Inputs:
- $I_s \in \mathbb{R}^{w\times h\times 3}$ source RGB image
- $I_t \in \mathbb{R}^{w_t\times h_t\times 3}$ target RGB image
- $R_s \in \mathbb{R}^{w\times h}$ ROI image
- $\mathbf{w_k} \in \mathbb{R}^{w\times h},\ k=1,\dots,m$ weight functions

Outputs:
- $T_k^0 \in \mathbb{R}^{2\times 3},\ k=1,\dots,m$ affine transformations

```
// SIFT points
// other feature point detectors should produce similar results
S_s ← SIFT(I_s | R_s > 0)                      // cv::FeatureDetector
S_t ← SIFT(I_t)                                // cv::FeatureDetector
(M_s, M_t) ← match(S_s, S_t)       // ratio test (0.8) + mutually best
T ← delaunay(M_s)
remove nodes from T with triangle flips in T(M_t)
compute piecewise affine map M_SIFT : T(M_s) → T(M_t)
compute global affine map A using RANSAC on (M_s, M_t)
Solve                                          // cv::solve, γ ≈ 0.1
```

$$\arg\min_{T_k^0,\ k=1,\dots,m} \sum_{\mathbf{p}\in\mathcal{T}} R_s(\mathbf{p})\|\mathcal{M}_{\text{SIFT}}(\mathbf{p}) - \sum_{i=1}^m w_k(\mathbf{p})T_k^0\mathbf{p}\|^2 + \gamma\sum_{\mathbf{p}\in I_s} R_s(\mathbf{p})\|A\mathbf{p} - \sum_{i=1}^m w_k(\mathbf{p})T_k^0\mathbf{p}\|^2$$

Algorithm 3: Warp Optimization in LBS Subspace

Inputs:

$I_s \in \mathbb{R}^{3wh}$	source RGB image vectors
$I_t \in \mathbb{R}^{3w_t h_t}$	target RGB image vectors
$\mathbf{w_k} \in \mathbb{R}^{wh},\ k = 1, \ldots, m$	weight functions
$T_k^0 \in \mathbb{R}^6,\ k = 1, \ldots, m$	initial affine transformations

Outputs:

$\mathcal{M} : R_s \to I_t$	warp from ROI to target
$\mathbf{a}, \mathbf{b} \in \mathbb{R}^m$	appearance parameters

// precomputation for warp

$G \in \mathbb{R}^{6wh} \leftarrow \text{gradient}(I_s)$ *// color gradient images in x and y*

// compute Jacobian

$J \in \mathbb{R}^{6wh \times 6m} \leftarrow 0$

for $k = 1, \ldots, m$ **do**

$$J(j,k) = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix} w_k(\mathbf{p}_j),\ \forall \mathbf{p}_j \in I_s$$

$J(j + 2wh, k) = J(j + wh, k) = J(j, k)$

end

// basic steepest descent images for each color channel

$\mathbf{SD} \in \mathbb{R}^{3wh \times 6m} \mid \mathbf{SD}(i,k) = G(2i-1:2i)^T J(2i-1:2i, k)$

// Block Hessians $H_i \in \mathbb{R}^{6m \times 6m}$, with 10×10 blocks B_i.

// $N_i = 3(\#\ pixels)$ and $L = \#\ blocks$

$H_i \leftarrow \sum_{\mathbf{p}_j \in \mathbf{B}_i} \mathbf{SD}(j,:)^T \mathbf{SD}(j,:),\ i = 1, \ldots, L$

$T_k \leftarrow T_k^0,\ k = 1, \ldots, m$ *// initial warp parameters*

// precomputation for appearance

$\mathbf{A} \in \mathbb{R}^{3wh \times 2m} \mid \mathbf{A}(:,k) = \begin{pmatrix} w_k.{*}I_s & w_k \end{pmatrix}$ *// appearances*

// Block appearance Hessians $H_i^a \in \mathbb{R}^{6m \times 6m}$

// same blocks as in the block Hessians

$H_i^a = \sum_{\mathbf{p}_j \in \mathbf{B}_i} \mathbf{A}(j,:)^T \mathbf{A}(j,:),\ i = 1, \ldots, L$

$\boldsymbol{\lambda} \leftarrow 0$ *// initial appearance parameters*

// Main loop

repeat

$Z \leftarrow \text{backwards\_warp}(I_t, \mathcal{M})$

$E \leftarrow Z - I_s - \sum_{i=1}^{2m} \lambda_i A_i(:,i)$ *// difference image*

$R = \phi'(E(\mathbf{p})^2)$ *// robustness image*

$\phi_i' \leftarrow (\sum_{\mathbf{p} \in \mathbf{B}_i} R(\mathbf{p}))/N_i$ *// average block robustness values*

$H_\phi^a \leftarrow \sum_{i=1}^{L} \phi_i' \cdot H_i^a$ *// update robust appearance Hessians*

$\Delta\boldsymbol{\lambda} \leftarrow H_\phi^a \setminus \left[\mathbf{A}^T (R.{*}E)\right]$ *// solve linear system*

$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$ *// update Appearance parameters*

$E \leftarrow Z - I_s - \sum_{i=1}^{2m} \lambda_i A_i(:,i)$ *// update difference image*

$R = \phi'(E(\mathbf{p})^2)$ *// update robustness image*

$\phi_i' \leftarrow (\sum_{\mathbf{p} \in \mathbf{B}_i} R(\mathbf{p}))/N_i$ *// average block robustness values*

$H_\phi \leftarrow \sum_{i=1}^{L} \phi_i' \cdot H_i$ *// update robust Hessian*

$\Delta T_k,\ k = 1, \ldots, m \leftarrow H_\phi \setminus \left[\mathbf{SD}^T (R.{*}E)\right]$

// Update warp

$\mathcal{M}(\mathbf{p}) \leftarrow \sum_{k=1}^{m} w_k(\mathbf{p}) T_k \left(\Delta T_k\right)^{-1} \mathbf{p},\ \forall \mathbf{p} \in I_s$

until $\varepsilon > |\Delta \mathbf{T}_k|,\ k = 1, \ldots, m$

// Appearance (gain and bias) parameters

$\mathbf{a} = \boldsymbol{\lambda}(1:2:end-1)$

$\mathbf{b} = \boldsymbol{\lambda}(2:2:end)$