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

Here, we provide pseudocode implementations for algorithms described in Transfusive Image Manipulation.

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

// seed selection
$G \leftarrow$ gradient_magnitude(I_s) + gradient_magnitude$(R_s == 0)$
// blur gradient
for $t = 1, \ldots, 5$ **do**
  $G \leftarrow G +$ conv(G, gaussian_kernel$(\sigma \approx 9)$)
end
confidence image $C \leftarrow R_s$.* $(1 - G)$
handles $H \leftarrow \{\}$
for $k = 1, \ldots, m$ **do**
  $H \leftarrow H \cup \arg\max_{\mathbf{p}} C(\mathbf{p})$
  $C \leftarrow C -$ gaussian_kernel$(\mu = \mathbf{p}, \sigma = \sqrt{|R_s > 0|/(\pi m)})$
end

// weight computation
$\Omega \leftarrow$ triangulate$(R_s > 0)$
for $\mathbf{v} \in \Omega$ **do**
  $\mathbf{v} \leftarrow (x, y, sI_s(x, y, 1), sI_s(x, y, 2), sI_s(x, y, 3))$ *// $s \approx 16$*
end
// Compute Q
$(L, M) =$ computeLM(Ω)
$Q \leftarrow LM^{-1}L$
for $k = 1, \ldots, m$ **do**
  Solve using MOSEK:

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

end
// enforce partition of unity
for $k = 1, \ldots, m$ **do**
  $\mathbf{w}_k(\mathbf{p}) \leftarrow \dfrac{\mathbf{w}_k(\mathbf{p})}{\sum_{\ell=1}^{m} \mathbf{w}_\ell(\mathbf{p})}$ $\forall \mathbf{p} \in \Omega$
end

Function computeLM(Ω)

// Compute L and M, for given mesh Ω
for *Triangle* (i, j, k) *in* Ω **do**
  $l_{ij} \leftarrow \|\mathbf{v}_i - \mathbf{v}_j\|, \quad l_{jk} \leftarrow \|\mathbf{v}_j - \mathbf{v}_k\|, \quad l_{ki} \leftarrow \|\mathbf{v}_k - \mathbf{v}_i\|$
  $r \leftarrow \frac{1}{2}(l_{ij} + l_{jk} + l_{ki})$ *// semi-perimeter*
  $\mathcal{A}_{ijk} \leftarrow \sqrt{r(r - l_{ij})(r - l_{jk})(r - l_{ki})}$ *// area*
  $\cot_{ij} \leftarrow \frac{1}{4}\left(l_{jk}^2 + l_{ki}^2 - l_{ij}^2\right)/\mathcal{A}_{ijk}$
  $\cot_{jk} \leftarrow \frac{1}{4}\left(l_{ki}^2 + l_{ij}^2 - l_{ki}^2\right)/\mathcal{A}_{ijk}$
  $\cot_{ki} \leftarrow \frac{1}{4}\left(l_{ij}^2 + l_{jk}^2 - l_{ki}^2\right)/\mathcal{A}_{ijk}$
  for *each of 6 permutations* (a, b, c) *of* (i, j, k) **do**
    $L(a, b) \leftarrow L(a, b) - 0.5 * \cot_{ab}$
    $L(a, a) \leftarrow L(a, a) + 0.5 * \cot_{ab}$
    if $\cot_{ij} \ge 0$ and $\cot_{jk} \ge 0$ and $\cot_{ki} \ge 0$ **then**
      $M(a, a) \leftarrow M(a, a) + \frac{1}{8} l_{ab}^2 \cot_{ab}$
    else
      $M(a, a) \leftarrow M(a, a) + \begin{cases} \frac{1}{8}\mathcal{A}_{ijk} & \text{if } \cot_{bc} \ge 0 \\ \frac{1}{4}\mathcal{A}_{ijk} & \text{otherwise} \end{cases}$
    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
$\mathbf{w_k} \in \mathbb{R}^{w \times h},\ k = 1, \ldots, m$ weight functions
Outputs:
$T_k^0 \in \mathbb{R}^{2 \times 3},\ k = 1, \ldots, m$ affine transformations

// SIFT points
// other feature point detectors should produce similar results
$S_s \leftarrow$ SIFT$(I_s \mid R_s > 0)$ *// cv::FeatureDetector*
$S_t \leftarrow$ SIFT(I_t) *// cv::FeatureDetector*
$(M_s, M_t) \leftarrow$ match(S_s, S_t) *// ratio test (0.8) + mutually best*
$\mathcal{T} \leftarrow$ delaunay(M_s)
remove nodes from $\mathcal{T}$ with triangle flips in $\mathcal{T}(M_t)$
compute piecewise affine map $\mathcal{M}_{\text{SIFT}} : \mathcal{T}(M_s) \to \mathcal{T}(M_t)$
compute global affine map A using RANSAC on (M_s, M_t)

Solve *// cv::solve, $\gamma \approx 0.1$*

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

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:

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

// precomputation for warp

$G \in \mathbb{R}^{6wh} \leftarrow$ 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

$Z \leftarrow$ 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

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

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