Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

Supplemental: Visualizing the Uncertainty of Graph-based 2D Segmentation with Min-path Stability

B. Summa[1], J. Tierny[2], and V. Pascucci[3]

[1]Tulane University
[2]Sorbonne Universites, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606
[3]Scientific Computing and Imaging Institute, University of Utah

1. Pseudocode of Major Algorithms

Algorithm 1 Live-wire Min-path Stability

```
Input: constraints: C; trees: 𝒯; graph: 𝒢
Output: 𝒮
for all updated c_i ∈ C do
    compute 𝒯_i
end for
for all p ∈ 𝒢 do
    𝒮(p) ← ∞
    for all pairs of consecutive c_i, c_j ∈ C do
        𝒮^{ij}(p) ← cost(p, 𝒯_i) + cost(p, 𝒯_j)
        𝒮(p) ← min(𝒮(p), 𝒮^{ij}(p))
    end for
end for
```

Algorithm 2 Graph Cut Smoothness Min-path Stability

```
Input: annuli: F; trees: 𝒯; graph: 𝒢
Output: 𝒮_s
for all f^i ∈ F do
    𝒩^i, 𝒩̂^i ← splitAndReplicate(f^i)
    for all n^i_j ∈ 𝒩^i and n̂^i_j ∈ 𝒩̂^i do
        compute 𝒯^i_j and 𝒯̂^i_j
    end for
end for
𝒮_s(p) ← ∞
for all p ∈ 𝒢 do
    for all f^i ∈ F do
        𝒮^i_s(p) ← ∞
        for all n^i_j ∈ 𝒩^i and n̂^i_j ∈ 𝒩̂^i do
            𝒮^i_s(p) ← min(cost(p, 𝒯^i_j) + cost(p, 𝒯̂^i_j), 𝒮^i_s(p))
        end for
    end for
    𝒮_s(p) ← min(𝒮^i_s(p), 𝒮_s(p))
end for
```

Algorithm 3 Alternative Minimum Paths

```
Input: queue: Q; trees: T; graph: G
Output: AP
procedure COUNT(p, i)
    path = path(p, T_i)
    υ, ω ← 0
    for all n ∈ path do                                   ▷ p to root
        if V_i(n) then
            (υ, ω) ← (υ, ω) + B_i(n)
            return (υ, ω, path, n)
        end if
        if n ∈ AP then ω ← ω + 1 else υ ← υ + 1 end if
    end for
    return (υ, ω, path, n)
end procedure
procedure REJECT(path, n, i)
    (υ, ω) = B_i(n)
    for all m ∈ path do                ▷ n to p, skip n unless n is root
        if n ∈ AP then ω ← ω + 1 else υ ← υ + 1 end if
        B_i(m) = (υ, ω), V_i(m) = True
    end for
end procedure
procedure FIND
    Reset all V_i, V_j, B_i, and B_j
    for all p ∈ G do                        ▷ by ascending cost from Q
        (υ_i, ω_i, path_i, n_i) ← COUNT(p, i)
        (υ_j, ω_j, path_j, n_j) ← COUNT(p, j)
        if (ω_i + ω_j)/(υ_i + υ_j + ω_i + ω_j) ≤ β then
            AP = AP + (path_i ∪ path_j)
            if |AP| == k then return AP else FIND end if
        else
            REJECT(path_i, n_i, i), REJECT(path_j, n_j, j)
        end if
    end for
end procedure
```