At a Glance

Issue Tree representations [many algorithms [1, 2, 3]] + attributes (very few explicit computation [4]): popular tools in MM and IP.

Goal Proposition of some efficient algorithms for computation of attributes and saliency maps.

Contribution A set of efficient algorithms for computation of:
- some accumulated information on region, contour, and context;
- extremal information along the contour (e.g., NFA [5]);
- extinction-based saliency maps [6].

Basic structure and information to compute + some computation principles

Algorithm computing incrementally some accumulated information

1 UNION_FIND(R) // black lines: union-find process; gray lines: for information on contour
2 for all p do
3 zpar(p) ← undef;
4 Ap(p) ← 0; L(p) ← 0; X(p) ← 0; X(∗) ← 0; Vz(p) ← Mz(p) ← 0;
5 for all e do is_boundary(e) ← false;
6 for i ← n − 1 to 0 do
7 p ← ∇(i), parent(p) ← p, zpar(p) ← p;
8 Ap(p) ← Ap(p) + 1;
9 for all n ∈ N(p) such as zpar(n) ≠ undef do
10 r ← FIND_ROOT(zpar(n)); if r ≠ p then
11 parent(r) ← p, zpar(r) ← p;
12 A(p) ← A(p) + A(r); L(p) ← L(p) + L(r);
13 X(r) ← X(p) + X(r); X(∗) ← X(∗) + X(∗);
14 for all e ∈ E(p) do
15 if not is_boundary(e) then
16 L(e) ← L(p) + 1; ε(e) ← information on 1-faces
17 /\ e and e′: top-right and down-left context of 1-faces
18 if e is above or on the right of p then
19 X(e) ← X(p) + X′(e); X(∗) ← X(∗) + X′(e);
20 else X(e) ← X(p) + X′(e); X(∗) ← X(∗) + X′(e);
21 appear(e) ← p;
22 else ε(e) ← 0;
23 if is_boundary(e) then
24 L(e) ← L(p) − 1;
25 if e is above or on the right of p then
26 X(e) ← X(p) + X′(e); X(∗) ← X(∗) + X′(e);
27 else X(e) ← X(p) + X′(e); X(∗) ← X(∗) + X′(e);
28 vanish(e) ← p;
29 for all do
30 if appear(e), Np ← vanish(e);
31 while Np ≠ Np do
32 Vz(Np) ← update(Vz(Np), ε2(e)); /\ update: either min or max
33 Np ← parent(Np);
34 return parent(Np);
35

Algorithm computing exact contextual information

1 COMPUTE_SALIENCY_MAP(f)
2 (T, A) ← COMPUTE_TREE(f);
3 ε1 ← COMPUTE_EXTENSION(T, A);
4 for all do X(∗) ← 0;
5 for all do
6 Np ← appear(e), Np ← vanish(e);
7 while Np ≠ Np do
8 Mz(Np) ← max(Mz(Np), Mz(e)); Np ← parent(Np);
9 for all 0-face do Mz(e) ← max(Mz(e), Mz(e), Mz(e), Mz(e));
10 return Mz

Algorithm computing extremal information and saliency map

1 COMPUTE_EXTENSION(T, A)
2 for each ε1 do X(∗) ← 0;
3 for each do ε1 ← COMPUTE_REGION(ε1, A);
4 return ε1

Complexity analysis

Accumulated information on region

<table>
<thead>
<tr>
<th>Time complexity</th>
<th>Auxiliary memory cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(n α(n))</td>
<td>O(n α(n))</td>
</tr>
<tr>
<td>O(n α(n))</td>
<td>O(n α(n))</td>
</tr>
<tr>
<td>O(n α(n))</td>
<td>O(n α(n))</td>
</tr>
</tbody>
</table>

Union-find process: O(n α(n));
- n: number of points;
- h: height of the tree;
- ε: maximal distance to the contour.

Computation of exact contextual information

- Extremal information along contour
- Extinction-based saliency map

<table>
<thead>
<tr>
<th>Time complexity</th>
<th>Auxiliary memory cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>O(n ε h)</td>
<td>O(n h)</td>
</tr>
<tr>
<td>O(n h)</td>
<td>O(n h)</td>
</tr>
</tbody>
</table>

References