Tree of shapes $\mathcal{T}$ [1, 2]: a versatile tool for many applications


At a glance

Motivation
- Significant contours of objects ⇔ segments of level lines [1]
- Inclusion relationship ⇒ tree of shapes $\mathcal{T}$ [2]: a versatile representation
- The knowledge of tree structure is fundamental for a deep tree analysis

Problem
- The number of shapes is about as large as the number of pixels

Objective
- Select a subset of level lines representing the main tree structure

Contribution
- An efficient algorithm for extracting meaningful and disjoint level lines
- A simplified image providing an intuitive idea about main tree structure

Basic idea

Select a subset of meaningful and disjoint level lines from the tree of shapes $\mathcal{T}$ to represent the main tree structure; Two main ideas:
1. $\forall N \in \mathcal{T}$, find its lowest ancestor shape $N'$: Smallest Enclosing Shape $\text{ses}(N)$, such that $N \subseteq N'\cap \partial N' \cap \partial N = \emptyset$.
2. $\forall N \in \mathcal{T}$ in some Order if it is not deactivated by any descendant, and none of $[N \sim \text{ses}(N)]$ is selected, then deactivate $[N \sim \text{ses}(N)]$.

Algorithm overview: three main steps
1. Tree of shapes construction: use the union-find-based algorithm in [4] to compute the set of all level lines.
2. SES computation: bottom-up traversal updating based on the nodes’ depth
3. Level lines selection: sequential test based on the status of $[N \sim \text{ses}(N)]$

Smallest Enclosing Shape (SES) computation

The algorithm in [4] works on Khalimsky grid $\mathcal{K}_D$. A shape is represented by a 2-face; parent: inclusion relationship; getCanonical: canonical element.

```
COMPUTE_SES(parent, S, depth)
for each x in $\mathcal{K}_D$ do SES(x) ← getCanonical(x)
for each 2-face x in reverse order of S do
    for each 0 and 1-face in x do
        if depth(e) < depth(SES(x)) then
            SES(x) ← getCanonical(e)
        else
            SES(x) ← SES(x)
    SES(q) ← SES(x)
return SES
```

References