Pacific Graphics 2025
M. Christie, N. Pietroni, and Y.-S. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 7

Supplementary Material for: Single-Line Drawing Vectorization

Tanguy Magne and Olga Sorkine-Hornung

ETH Zurich, Switzerland

1. Graph traversal algorithm

The following algorithms describe the graph traversal presented in Section 3.3.2 of the main paper.

Algorithm 1: Graph Ordering

Input : Simplified medial axis graph $\mathcal{G} = (V, E)$ with classified nodes, list of endpoints $\mathcal{E}$
Output : Set of strokes (lists of nodes) representing the drawing
Initialize : $strokes \leftarrow GraphTraversal(\mathcal{G}, \mathcal{E})$;
`// Returns if single stroke`
if *strokes* contains a single stroke **then**
  return *strokes*
end
`// Flip low confidence degree-4 nodes until a single stroke is produced`
$to\_flip \leftarrow$ all low confidence degree-4 nodes common to 2 strokes;
$i \leftarrow 0$;
$best\_strokes \leftarrow strokes$;
while $i < len(to\_flip)$ **do**
  $flip\_intersection(\mathcal{G}, to\_flip[i])$;
  $strokes \leftarrow GraphTraversal(\mathcal{G}, \mathcal{E})$;
  if *strokes* contains a single stroke **then**
    return *strokes*
  end
  if $len(strokes) < len(best\_strokes)$ **then**
    $best\_strokes \leftarrow strokes$;
  end
  $i \leftarrow i + 1$;
end
`// Return the configuration with the lowest amount of strokes`
return *best_strokes*

Algorithm 2: Graph Traversal

Input : Simplified medial axis graph $\mathcal{G} = (V, E)$ with classified nodes, list of endpoints $\mathcal{E}$
Output : Set of strokes (lists of nodes) representing the drawing
Initialize : $strokes \leftarrow \{\}$;
set all the nodes in $\mathcal{G}$ to unvisited state;
while $len(\mathcal{E}) > 0$ **do**
  add $TravelGraphStartingFrom(\mathcal{G}, \mathcal{E}[0], \mathcal{E})$ to *strokes*;
  remove the first and last node of the stroke from $\mathcal{E}$;
end
`// While not all nodes have been visited, add new strokes starting from random degree-2 nodes`
while not all nodes visited **do**
  $start\_from \leftarrow$ random degree-2 node from $\mathcal{G}$;
  add $start\_from$ to $\mathcal{E}$;
  add $TravelGraphStartingFrom(\mathcal{G}, start\_from, \mathcal{E})$ to *strokes*;
end
return *strokes*

Algorithm 3: Get Initial Direction

Input : Simplified medial axis graph $\mathcal{G} = (V, E)$ with classified nodes, endpoint *start* to start the traveling on the graph
Output : The next node to travel to
if $deg(start) = 1$ **then**
  next $\leftarrow$ the sole neighbor of *start*;
end
else if $deg(start) = 2$ **then**
  next $\leftarrow$ any unvisited neighbor of *start*;
end
else if $deg(start) = 3$ **then**
  next $\leftarrow$ direction opposite to largest angle between tangents;
end
return *next*

Algorithm 4: Travel Graph Starting from

```
Input      : Simplified medial axis graph 𝒢 = (V, E) with
             classified nodes, endpoint start to start the
             traveling on the graph, list of endpoints ℰ
Output     : A stroke represented by a list of nodes
Initialize : stroke ← [ ]
add start to stroke and mark start as visited;
// Determine initial direction
current ← GetInitialDirection(𝒢, start);
previous ← start;
terminate ← false;
// Main traversal loop
while not terminate do
    if current in ℰ then
        terminate ← true;
    end
    switch deg(current) do
        case 1 do
            next ← the sole neighbor of current;
        end
        case 2 do
            next ← the neighbor that is not previous;
        end
        case 3 do
            if current is Y-junction then
                if degree-1 neighbor not visited then
                    next ← degree-1 neighbor;
                end
                else
                    next ← the sole unvisited direction;
                end
            end
            else
                // current is in ℰ
                if not all neighbors traversed then
                    terminate ← false;
                    next ← direction of closest tangent;
                end
            end
        end
        case 4 do
            next ← node according to CNN prediction;
        end
    end
    add current to stroke and mark current as visited;
    previous ← current;
    current ← next;
end
return stroke
```