Shape Detection Algorithm using Cycle Detection (DFS)

April 21, 2017

1 Algorithm

Algorithm is divided into 2 functions.

Algorithm 1 Shape Detection algorithm

procedure GETCLOSEDPOLYGON($line\_list$) ▷ line_list is an arroy of line segments
    $polygon\_list \leftarrow []$ ▷ List of polygon identified
    $count \leftarrow 0$ ▷ Count the number of iteration of loops
    while line_list is not empty and $count \neq len(line\_list)$ **do** ▷ This loop choses first line of the line_list and find if it is a part of a cycle or not
        $count \leftarrow count + 1$
        $visited \leftarrow []$ ▷ visited bool array initialized to false
        $first \leftarrow line\_list[0]$ ▷ first line in line_list
        $vertices \leftarrow [first.x1, first.y1], [first.x2, first.y2]$ ▷ Initialize a list of vertices for tracing a polygon
        $visited[0] \leftarrow False$ ▷ Marking first line visited
        $check, vertices =$ RECURPOLYGON($line\_list, vertices, visited$)
        if $check == True$ **then**
            $polygon\_list.append(vertices)$ ▷ Append newly found polygon
            $line\_list.remove(vertices)$ ▷ Delete detected lines from line_list
        else ▷ Remove the first line and push it in back
            $temp = line\_list.pop(0)$
            $line\_list.append(temp)$
    return $polygon\_list, line\_list$ ▷ line_list contains remaining lines

Algorithm 2 Cycle Detection Recursion Function (DFS)

```
1: procedure RECURPOLYGON(line_list, vertices, visited)
2:     next_vertex ← vertices[−1]          ▷ Last point of vertices for matching
3:     second_check ← vertices[−2]     ▷ Second last point for same line check
4:     for each line in line_list do
5:         v1 ← [line.x1, line.y1]   ▷ Possbility of any two end points to match
6:         v2 ← [line.x2, line.y2]
7:         if next_vertex == v1 and second_check ≠ v2 then
8:             if visited[line] == False then      ▷ If match found and it's not
   visited
9:                 visited[line] ← True                        ▷ Mark the line visited
10:                vertices.append(v2)
11:                if vertices[−1] == vertices[0] then  ▷ If first and last points
   are same, then cycle is found
12:                    return True, vertices
13:                check, list_till = RECURPOLYGON(line_list, vertices, visited)
14:                if check == True then     ▷ If recursive function gives True,
   then that means the path made a complete cycle
15:                    return True, list_till
16:                else                                          ▷ Backtracking
17:                    vertices.pop()
18:            else
19:                return False, empty
20:        if next_vertex == v2 and second_check ≠ v1 then
21:            if visited[line] == False then   ▷ Similar steps as previous one
22:                visited[line] ← True
23:                vertices.append(v1)
24:                if vertices[−1] == vertices[0] then
25:                    return True, vertices
26:                check, list_till = RECURPOLYGON(line_list, vertices, visited)
27:                if check == True then
28:                    return True, list_till
29:                else
30:                    vertices.pop()
31:            else
32:                return False, empty
33:    return False, empty ▷ If nothing match is found, then it will not make
   a cycle
```