Formal description of Thesis problem

Raphaël P

18/12/2014

1 Description

The aim of the project is to build Nonograms (of different difficulties) using curves as opposed to lines. The curves to be used are Bézier curves. A section of the curve is defined by 4 points, two being the end points of the section and the other two being the direction in which the curves go.

I shall start with using polygonal lines instead of Bézier curves, as they are easier to work with. As a matter of fact, looking at straight lines, with four different slopes: vertical line, horizontal line, diagonal + line, diagonal - line. Diagonal lines have slopes of 45°

1.1 Local Operations

These are the operations used to either extend a curve to the bounding box or glue it to some other curve, in order to make the Nonogram more complex/ interesting.

2 Input

Input will either be a set of points defining the original picture (stored in an external file) or a set of polylines defining the picture. The polylines/ points do not have to represent the picture exactly, we are looking for an approximation.

The input file can be a set of coordinates representing multiple loops, where a loop is a polygon or set of polygon with the same depth. The depth is whether or not the polygon is part of the original picture. We consider the depth because of holes. A polygon with a hole, itself containing a polygon can be thought of a sequence of three loop. The main loop, (loop 1) consists of the coordinates of the main polygon, it has depth 1. The second loop is the hole present within the main polygon, it has depth 2 and loop 2 is the set of coordinates of the hole. The last polygon, within the hole but part of the original picture has depth 3 and its loop is the set of coordinates defining it. With this definition, all odd numbered depth polygons are part of the original drawing and even numbered depth polygons are holes (not part of the original drawing). FIND A WAY TO REPRESENT THOSE. (maybe a set of coordinates with some indication for the depth of the current polygon)

Input file can also be an 'svg' file. Those are XML/HTML files with information about the shapes. Here is a svg file for a polygon with 5 sides:

```
<!DOCTYPE html>
<html>
<body>

<svg height="400" width="500">
  <polygon points="220,10 300,210 250,400 170,250 123,234"
 style="fill:lime;stroke:purple;stroke-width:1" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

We can work on python with svg files, using the svgwrite module. Already installed using pip. Here are the basic functionalities:

resources:https://pythonhosted.org/svgwrite/

```
import svgwrite
dwg = svgwrite.Drawing(filename='test.svg',
size=("800px","600px"))
# main function to add to the current svg fill defined above
dwg.add(dwg.polygon(points=[(x,y),(x,y),...],
stroke_width="1",
stroke="black" # can also use rbg
fill_opacity = "0.0" # how opaque (0.0 to 1.0)
fill"blue" # filling colour))
dwg.save()
```

Using these could be beneficial because you can already tell the shape represented from the header, 'polygon',

3 Output

Output is a Nonogram (of a certain difficulty) based on curves (polygonal straight lines at the beginning).

4 Conditions

Definition of Nonogram:

- a set of four lines defining a bounding box
- a picture embeded in a set of lines/curves
- for each line/curve a description: the number of faces to colour on each of its sides. Using a comma to delimit coloured faces from non-coloured faces. Example: 1,2 means that from the bounding box, 0 or more faces should be left blank, then 1 face should be coloured, then 1 or more faces should be left blank, then 2 faces should be coloured and then 0 or more faces should be left blank to the other side of the bounding box. Using regular expression syntax: $0^{\star}1^{1}0^{+}1^{2}0^{\star}$, where 0 are blank faces and 1 are coloured faces, and the superscript shows the number of faces to colour ($\star$ is zero or more and $+$ is one or more)

- Several conditions must be met:

  - Fatness (section 4.1) of facets should be somewhat similar. We look at fatness as opposed to area as we want the general shape of the faces to be somewhat normal.
  - Facets geometry should be similar (depends, as some weird looking facets will be quite interesting too) Dont want long and narrow facets as it can be ambiguous with regards to other facets.
  - Facets 'area' should be similar (up to a constant factor) We do not want small areas
  - Intersection points should be far away enough from one another (section 4.2)
  - do we allow for 3/4 line intersections? With only 4 different slopes, we can obtain at most a four line intersection. The problem then becomes to know which face is adjacent to the intersection: the edge adjacent faces (only 2) or the vertex adjacent faces (all 4). ASK PEOPLE FOR WHAT THEY THINK.
  - There should be a restriction on the angle at which two lines intersect(to be determined and not needed for the 4 slopes version)
  - We dont want curves/lines that do not do 'anything', namely each curve should cross the pictogram at least once
  - We might want to curves/lines to be somewhat uniformly distributed in the bounding box
  - the Nonogram should be uniquely solvable (only one image can results from solving it)
  - the Nonogram should reflect the input difficulty (4.3)
  - From the bare Nonogram, the final image should not be visable.

4.1 Fatness

The Fatness of a facet can be described in several way. One of them is to look at the closest enclosed circle (radius r) and the smallest bounding circle (radius R) and compute the fatness as the ration $\frac{r}{R}$

4.2 Intersection distance

how far away should intersection points be. Variable using the programme to see what kind of Nonogram are obtained depending on its value.

4.3 Difficulty of Nonogram