SELECTED ENGINEERING PROBLEMS
NUMBER5
INSTITUTE OF ENGINEERING PROCESSES AUTOMATION
AND INTEGRATED MANUFACTURING SYSTEMS

Krzysztof FOIT

Institute of Engineering Processes Automation and Integrated Manufacturing Systems
krzysztof.foit@polsl.pl

SVG GRAPHICS LANGUAGE AS A DESCRIPTION OF A 2D PATH IN ROBOT PROGRAMMING TASKS

Abstract: In this paper, the issue of programming of the complex path of robot effector will be discussed. It should be noted that the most of accurate tracks with complex shape are implemented for operations like milling, deburring, applying seals, adhesives, etc. These tasks are usually performed on the plane. Programming of these operations usually requires a lot of effort – operator should indicate successive points that are necessary to define the path of movement. Using dedicated software, greatly simplifies this task, but its purchase is often associated with considerable costs and attachment to a particular platform. The solution, proposed in this paper, uses the SVG graphics to define the effector's path. It is based on open standards that simplify implementation of the method on a variety of hardware platforms. SVG files are also easy to process, because of its textual form that refers to the XML.

1. Introduction

In addition to the transport and manipulative tasks, industrial robots are often used in the technological process to carry out such tasks as welding, deburring, applying adhesives and sealants, etc. [1,2]. A common feature that connects these operations is a large degree of complexity of the tool path and the fact that the individual operations are often carried out in the same plane.

Due to the considerable complexity of the path, the robot is often programmed using CP (Continuous Path) trajectory planning, or – in the case of less complex path – PTP (Point-To-Point) trajectory planning. Both methods require a fairly significant effort, amount of time and – in the case of planning CP – direct interaction with the robot.

Some attempts are being made to simplify the procedure of programming of complex paths. This is particularly noticeable in the case of welding robots, due to the frequency of use. The literature [3-7] contains descriptions of special graphic interfaces or programming languages intended for rapid prototyping of a robot program. The manipulators are also equipped with special vision systems, which allow the real-time adjustment of program parameters, according to the environmental conditions. These solutions are often dedicated to a particular type of robot and the use of vision systems significantly increases the overall cost of implementation.

One of the ways to reduce the cost of deploying new software is to use the applications under a free license, or open standards. Large companies – where robotized production lines are a kind of standard – decide not to use such software or open standards, in fear of improper level of assistance. It seems to be the most important reason for using closed solutions, supplied by machine manufacturers. However there is a middle ground: the use of open standards that are supported by major software vendors. Such standards include, inter alia, the XML markup language and SVG graphics standard that is the derivative of the XML.

The SVG was earlier used in the field of robotics as a part of the control interfaces [4-6], but is rarely used as a key part of the programming system [7-8].

This paper presents the concept of using SVG graphic to describe the path of tool mounted on the robot wrist.

2. The SVG standard

The SVG acronym stands for "Scalable Vector Graphics". Although the name suggests a graphic standard, SVG is a programming language that operates on the graphics elements of the drawing. In addition to the typical description of attributes, it also allows to make the transformations (including translation, rotation and scale) or even animations, which in turn allows creating complex patterns that are based on the definitions of simple figures. Another quite important feature of the SVG standard is that the file, which stores image information, has the text form and drawing can be edited using an ordinary, plain text editor.

The SVG standard uses vector representation of a drawing. It means that every geometric entity is treated as the object, not as a set of pixels. This makes the objects easy to manage, however can cause some problems during the transformation of drawing into a bitmap.

The development of the SVG standard started in 1998, in order to unify web vector graphics standard. In that time, several competing formats have been developed and all of them were revised by W3C group (World Wide Web Consortium), which is an international community that develops standards used in the World Wide Web. The SVG is based on the XML language (Extensible Markup Language), so it inherits its text format and all other dependencies [9]. This property is a very important advantage, because the file can be embedded into other XML documents, databases, HTML etc. The SVG can handle three types of objects: vector graphics, raster graphics and text.

The declaration of the SVG in the documents is similar to the declaration of the XML object (Fig. 1)

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    .......
        Graphics_objects
    .......
</svg>
```

Fig.1. The declaration of the SVG root object

In case of SVG, the root object has the form of <svg> tag. The "xmlns" variable is required by some browsers in order to correctly handle SVG graphics.

The SVG standard defines seven types of shapes: a rectangle, a circle, an ellipse, a line, a polygon, a polyline and a path. Additionally there is a special shape named "text", which allows placing a text in the drawings. It differs from the other objects, because a text is defined as letters, words, and sentences – not as geometrical shapes.

Every SVG object and its attributes use descriptive names, so in many simple cases there is no need to use graphics editor in order to create or edit a simple drawing. As a result the user could use any simple text editor to create or edit a document. In many cases this is enough to make minor corrections in a code. It is a major advantage, when we taking into consideration the cooperation with the robot's controller, using simple computer or other IT equipment.

It is worth to mention about the SVG "path" object and describe it in a few words. Its definition covers a wide variety of graphics shapes, and selection of the proper shape is done by setting values of the object's attributes. These attributes control the movement of the virtual pen within the borders of the canvas as well as define complicated shapes, like cubic or quadratic Bezier curves. It is done by placing a special, one letter commands in the attribute's value string. The command could be written in upper- or lowercase. The uppercase refers to the global coordinate system, while lowercase initiates operation in relation to the current position of the pen. The most important commands, selected from the set of all "path" object attributes, are:

- M – moves the pen to the specific point,
- L – draws a line,
- A – draws an arc,
- Q – creates the quadratic Bezier curve,
- C – creates the cubic Bezier curve,
- Z – closes the path.

Other commands extend the previously listed ones. They relate to the special definitions of the objects, like cubic and quadratic Bezier curves or horizontal/vertical lines. They are as follows:

- H – draws horizontal line,
- V – draws vertical line,
- S – draws smooth cubic Bezier curve,
- T – draws smooth quadratic Bezier curve.

All of the mentioned commands are the subsystem of the "path" object that have considerable drawing capabilities and could replace almost all of remaining SVG graphics objects, like circle, polyline or rectangle. There is no objections to create the SVG file in a text editor, where the simple objects (like circle, rectangle etc.) could be described by editing the SVG code. On the other hand, in case of complex paths, it is recommended to use an SVG editor. The example of the code that uses the "path" object and the result of its interpretation are shown in Figure 2.

```
<path
      style="fill:none;stroke:#000000;stroke-width:1px;stroke-linecap:butt;stroke-linejoin:miter;stroke-opacity:1"
      d="m 122.85714,75.219325
        c 75.08659,0.0826 150.20789,-0.151479 225.26786,0.08929 20.23364,2.007881 45.10569,3.672217 57.8064,21.881452 9.40607,
        19.717187 -7.1669,39.479067 -19.18946,53.924387 -38.53147,45.11078 -95.4723,70.40031 -130.80322,118.53078 -11.34598,
        13.97374 -16.28411,38.21239 1.85425,49.08031 30.22875,18.009 67.08452,17.2516 101.16141,17.69474 51.16074,
        -2.2979 102.41756,-0.75292 153.62568,-1.20095 24.37808,0 48.75615,0 73.13423,0"
      id="path2997"
/>
```

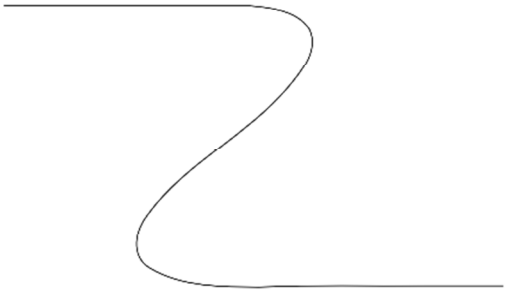

Fig.2. The example of "path" object declaration (cubic Bezier curve) and its graphical interpretation

The SVG file is saved as a XML text file with the ".svg" extension. It is also possible to save the file in a compressed form (using gzip deflate algorithm) with the ".svgz" extension. The compressed file is still readable by most programs, but the direct editing (using text editor) is no longer possible without the prior decompression.

The SVG files could also contain animated objects or hyperlinks, what makes them a very powerful tool among the World Wide Web standards.

All of the mentioned properties of SVG make this standard very attractive in a manner of creating user interfaces or graphics presentation. In the next part of the paper, it will be shown that this form of notation could be also used for development of robot's tool path.

3. Conversion of the SVG objects to the robot's tool path

The SVG standard, thanks to its versatility and wide processing capabilities, can be used to store information about the path of the tool mounted on the robot's wrist. Due to the SVG limitations, it can store information only about points on the plane. This is not a major disadvantage, because – as it was mentioned earlier – most of the tasks, which require high accuracy with a significant degree of path complexity, take place on a plane. The orientation of this plane (on which the task is realized), may require defining a new coordinate system, which is done by using the appropriate robot's control functions.

The transformation of the SVG path into the form that will be "understood" by the robot's controller may cause some difficulties. The paths or object, like circle or rectangle, should be transformed to a set of discrete points, which will describe the following segments of the tool path [8]. The transformation could be done in two manners:

- the vector graphics could be transformed to the bitmap, then the list of points, which contain every pixel that belongs to the SVG object, should be created,
- every objects should be sampled into the set of the discrete points, and every curve should be converted into segments.

The second case could be done more efficiently by using optimization. Some points on the straight segment of the path could be omitted, but on the curved sections the points must be

arranged more densely in order to achieve adequate accuracy. The optimization can cause some problems during the movement of the effector, because speed changes may occur. In order to avoid this problem, sometimes it is better to find the optimal step that will be compromise between changes of velocity and accuracy. One of the possible solutions of the sampling problems is to export the drawing to a HPGL file. This format was created especially for plotters, enabling the possibility to control a pen or cutting tool on the XY plane. Export to the HPGL format can be performed from any application that supports SVG, by using the print command, selecting the correct driver and the option to write to a file. Some applications, like Dia, Inkscape and some graphics converters, allow direct export to the HPGL. The both ways of getting the coordinates from the SVG path are illustrated in Fig. 3.

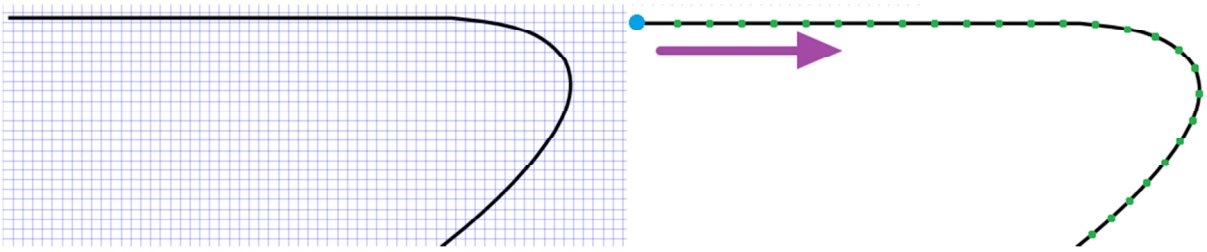

Fig.3. The rasterization (on the left) and sampling (on the right) of the SVG path

Regardless of the method used for acquisition of coordinates, the preparation of data for the robot program proceeds in a similar manner (Fig. 4). The preceding stage of the creation of the core of program is to arrange the appropriate coordinates into groups, corresponding to the individual objects (paths) of the SVG graphics. If the SVG drawing was exported to the HPGL file, this operation is quite simple, because the exporting algorithm arranges the coordinates automatically. In the case of rasterization, the whole bitmap must be scanned in order to acquire the coordinates of the points. The arrangement of the coordinates according to the objects requires more effort and special algorithms (e.g. nearest-neighbor search). Also excessive accuracy, resulting from the neighboring arrangement of bitmap's pixels, could cause some problems. It could results in robot's memory overload (due to the large number of coordinates) and excessively slow movement of the manipulator. It is therefore advisable to exclude some points from the set of pixels and optimize the resultant path..

4. Conclusion

The SVG is open standard and it is consistently evolving. This means that new features can be expected in the future releases of the SVG, together with growing support, documentation and software implementation. The use of SVG standard in robotics is very limited so far. Its main purpose is to create graphical user interfaces. The method presented in this paper changes the approach by introducing the ability to define manipulator path in graphical manner, using a flexible standard and widely available software tools. The advantages of using open standards are not only financial savings, but also the possibility of managing the code on different platforms, including the mobile devices, which are gaining more and more popularity today.

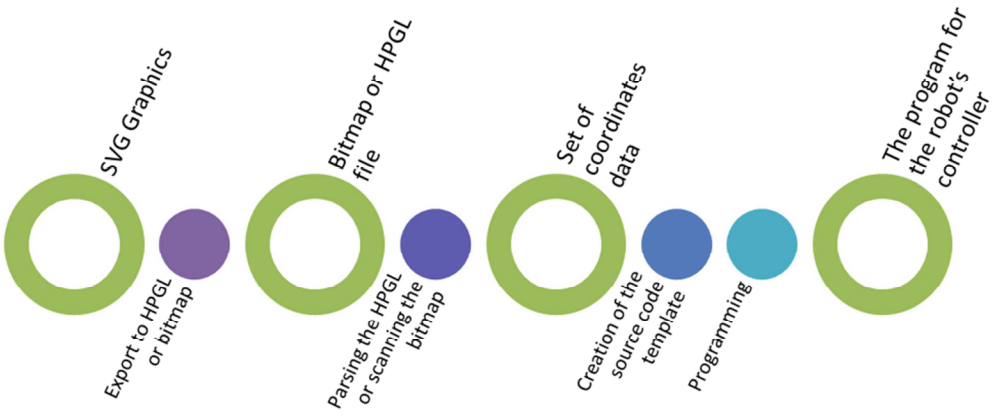

Fig.4. The process of preparation of the robot's program from the path designed in SVG

The main disadvantage of the presented method is its 2D nature, which means the inability to extend it to the general cases of robot's programming. On the other hand this drawback is not particularly problematic, since the most of precise actions performed by the robot, takes place on the plane.

Future work will focus on creation of a system, which will support the use of SVG graphics for planning of a manipulator path.

References

1. Ardayfio D.: Fundamentals of robotics, Taylor & Francis, 1987
2. Pires J.N., Loureiro A., Bölmsjo G.: Welding robots: Technology, system issues and applications, Springer 2006.
3. Dai W., Kampker M.: User oriented integration of sensor operations in a offline programming system for welding robots. In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on (Vol. 2, pp. 1563-1567). IEEE
4. Jo S., Shahab Q. M., Kwon Y. M., Ahn, S. C.: Indoor modeling for interactive robot service. In SICE-ICASE, 2006. International Joint Conference (pp. 3531-3536). IEEE
5. Jo S., Ki J., Jeon K. W., Kwon Y. M., Ahn S. C.: Human-Robot Interaction with Indoor Virtual Model. In Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on (pp. 493-498). IEEE
6. Reinicke C., Buss M.: Towards multi-modal mobile telepresence and telemanipulation, The Internet Challenge: Technology and Applications (Ed. Hommel G., Huanye S.), Springer, Netherlands, 2002, 55-62
7. Cassinis R., Tampalini F.: AMIRoLoS an active marker internet-based robot localization system. Robotics and Autonomous Systems, 55(4), 306-315.
8. Foit K.: Controlling the movement of the robot's effector on the plane using the SVG markup language, Advanced Materials Research, vol. 837 1662-8985, Trans Tech Publications 2014, pp. 577-581
9. Scalable Vector Graphics (SVG) 1.1 (Second Edition), W3C Recommendation 2011, 2011, [online] Retrieved June 25, 2013 from http://www.w3.org/TR/SVG/Overview.html