Using 2D Contours to Model Metal Sheets in Industrial Machining Processes

Aitor Moreno*, Álvaro Segura*, Harbil Arregui*, Jorge Posada*, Álvaro Ruíz de Infante, Natxo Canto****

* Vicomtech-IK4, Paseo Mikeletegui 57, San Sebastián 20009, Spain

** Lantek Sheet Metal Solutions, Parque Tecnológico de Alava - Albert Einstein 36, Miñano 01510, Vitoria, Spain

Abstract The cutting and punching industrial processes can be conceptualized as material removal methods, where the sheet is being transformed as the running CNC program is being executed by the NC machine. The simulation of such processes has to take into account this increasing complexity of the metal sheet. In this work we introduce a representation based on 2D contours to model the metal sheet in such industrial processes. All the operations of the programs are transformed to boolean operations between the sheet and the sweep of moving cylinder (cutting machines) or a complex polygon (for punching machines). As a direct application of the technique results in poor performance, a spatial subdivision has been used to increase the performance.

1 Introduction

In the machining industry, technological advances have led to obtain more efficient machinery, faster and more versatile, leading to better economical results. In the field of metal sheet cutting, the nature of the involved processes generates lot of wasted material, having to be returned to the melting industry for recycling.

Therefore, simulation tools can help to test and check the CNC programs in the design phase, being verified all the necessary times before they are actually run in the machines. After a series of tests, including the optimization modifications, the program can reach an optimized state, good enough to be transferred to production. Any not fully tested CNC program can cause or increase the risks and compromise the machinery, provoking partial breakages of parts, collisions between different machine parts, the metal sheet and the tool.

This work present some of challenges related to the representation of the metal sheet, found during the development of an interactive NC simulator for cutting and punching processes for metal sheets.

The cutting processes operate on the sheet, following the instructions provided by the CNC programs. They are composed of a large number of linear and arc movements, and other auxiliary instructions. All the machining operations can be conceptualized as boolean operations over the sheet. A linear or arc movement removes the material from the sheet following the path of the torch. Hence, a movement can be conceptualized as a sweep of a moving cylindrical tool.

In the punching processes, the material removed from the sheet is not a sweep of the tool, as a punch instruction only uses its profile to remove material only once. The challenges of the punching processes are related to the complexity of those patterns and the high frequency of such operations (up to 1000 punches per minute).

2 Related Work

The NC machining simulation using Computer Graphics techniques is a widely extended research topic, where the main issue is related to the representation of the dynamic parts of the simulation (in this case, the metal sheet is considered dynamic, as its geometry changes over time). Some traditional approaches do not store the geometrical information during the simulation, but they simply modify the drawing screen using an image-based approach.

Some techniques store the intermediate results in the computer's memory, having an internal 3D geometric representation of the object that is changed continuously during the simulation process. With these methods, a permanent representation is always available and provides free camera movement around the object, better geometric accuracy control, geometric based collision detection, etc.

Van Hook [3] used an extended Z-buffer data structure (called a Dexel structure) for the graphical verification. In his work, a scan method to convert surface data into his Dexel (depth element) structure was presented. The Z values for the nearest and the farthest surface at each Dexel are stored in such depth elements. This technique has been extended by several authors [12].

Other representation methods in the Computer Graphics field are fundamentally geometric like *i)* Boundary Representation (B-Rep) *ii)* Constructive Solid Geometry (CSG), and *iii)* Hierarchical Space Decomposition (HSP).

Although B-Rep is the most used method for solid modelling in modern CAD systems, its straightforward use for machining simulation is not convenient due to the long time required for the dynamic simulations [8]. A similar problem occurs with CSG representation, with computational costs of order $O(n^2)$, where n is the number of primitives [9] being computationally expensive. A modern implementation of CSG representation through BSP (Binary Space Partitioning) trees has been ported to Javascript and to the Web with a great but non real time performance [11].

To cope with the complexity of the problem and the long time required in these approaches, the approximation of the exact geometry, and especially the partitioning of the object in suitable regions has been proposed by several authors [9].

The most classic technique for volume partitioning is the voxel representation (classical octree, extended octree [1], SP-Octree [2]) that combines the space partitioning, solid representation and boolean operation support in a single definition. The sheet to be manufactured can be approximated by a very thin extruded plane, given that the machining program is limited to 2D movement over the sheet. This sheet representation provides a direct way to perform boolean operations between the moving tools and the planar sheet. The 3D boolean operation is simplified in a single 2D boolean operation between two 2D complex polygons, that is a well reviewed research topics [10, 7].

In this work, we present how an efficient and optimized metal sheet machining simulator has been designed and developed, with an internal geometrical core based on complex polygon boolean operations to support the material removal processes.

3 Punching and Cutting Simulator

A simulation system for cutting and punching of metal sheet through a NC controller was developed. The simulator is as a prototype simulation software module and it is focused to obtain consistent geometric results and high graphics quality for the following sheet machining processes (see **Fig** 1):

- Metal sheet cutting processes using laser, plasma, oxy-fuel or water jet.
- Punching processes with tools defined with basic or complex shapes.

By means of simulation techniques, the application emulates the behaviour of the machine tool to the computer. The simulation system takes as input a starting NC program (normally, G-code dialect) translated to a common and simplified XML format, listing all the movements that the machine will perform during the cutting or punching operation. The operation mechanisms differ between the cutting and punching process. The main significant difference is that in the cutting process the removal process is continuous while the torch is powered on, whereas in the punching process, the removal process is instantaneous when the punch is triggered.

In the following subsections, the system architecture and the main functionality will be described.

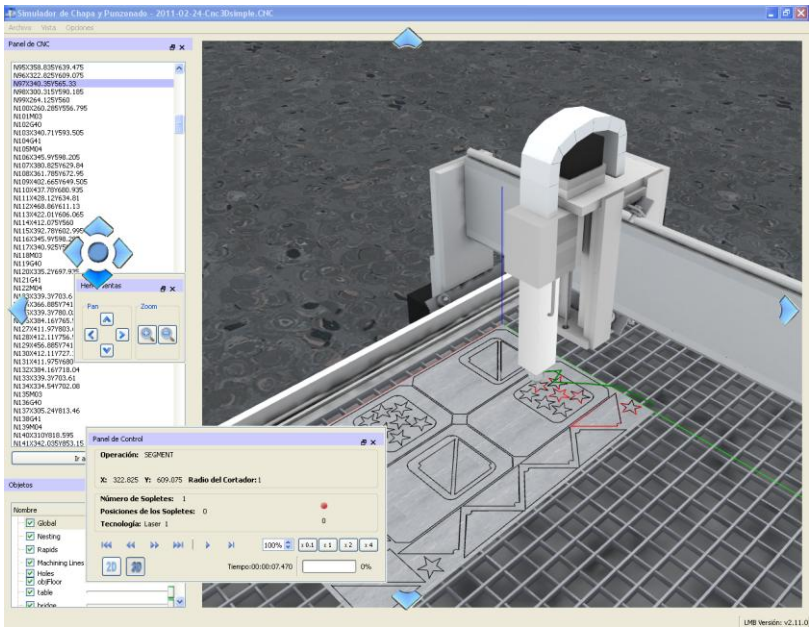

Fig 1. User Interface of the CNC simulator. The UI shows all the necessary information to the user (CNC instructions, simulation controls, visibility options and the 3D interactive output).

3.1 System Architecture

The simulator is structured as a multi-layered architecture, each one encapsulating different methods and techniques. The low level layer involves the geometric calculations (Geometric Kernel) with the management of the boolean operations between 2D polygons as its main responsibility. The Clipper library [4] provides the functionality to calculate boolean operations between 2D polygons, as the basis for subtraction between a complex polygon representing the sheet and another polygon representing the sweep of a moving tool in a given period of time (either cutting or punching).

The low level layer provides access to the graphic system, responsible for rendering the simulation results in the screen. OpenSceneGraph [6] was chosen to be our graphic engine, which is used to draw on screen the result of the boolean operations, the 3D machine models and all the virtual elements in the scene.

The middle layer of the architecture provides the logic behind the application domain. The concepts related to the NC machine domain, like *Part*, *Tool*, *Machine* and *Axis* are represented as classes. The animated behaviour of the moving elements is also represented. In this layer the geometric sweep volume of the moving

tools are calculated and passed to the lower level for the actual boolean operation with the metal sheet representation.

Over the cutting and animation layer, we have added the user oriented layer, which includes the graphical interface of the prototype application and management of the XML files. This has been necessary to:

- Provide the user interface, including multi-language interface options.
- Implement the different navigation methods in the 3D virtual world, using the mouse as input device.
- Manage and display the real NC instructions (highlighting the currently executing instruction) that will be loaded into the simulator through the intermediate and abstract XML representation.
- Manage and display the visualization options of the 3D objects corresponding to the individual parts of the cutting or punching machine.

Both the user interface and the XML parsing functionality have been implemented through the Qt library.

3.2 User Interface

The user interface (see **Fig** 1) displays a simple and clean interface between the user and the simulation functionality. It has been developed using the Qt library, which provides the technology needed to create professional and high-quality graphical interface controls. The application GUI is structured as a set of docking and floating panels, which can be moved freely or docked to any side of the screen with the 3D virtual world always in the central widget.

There are two ways of controlling the virtual simulation. One is by running a continuous simulation, where an animation factor is applied to control the simulation speed. The other is a fast mode, which will run the simulation in background as fast as possible, until the target instruction is reached. Combining both modes, users can go to a specific instruction of the NC code (shown in an independent panel) and from there, begin an animated simulation with the desired speed factor. The rest of the buttons in the Control Panel offer simple VCR functionality: play and stop, play only one instruction, and go directly to the previous or next instruction, the first or the last instruction of the loaded program.

The GUI provides the visibility options for all the objects in the virtual scene. Any object can toggle its visibility, but not all the elements can modify the transparency, as the lines (machining toolpaths and other helping elements) and the ground model. The camera properties and movements can be set up in another panel, including the toggle button to go to 3D or 2D mode and the zoom and pan functionality.

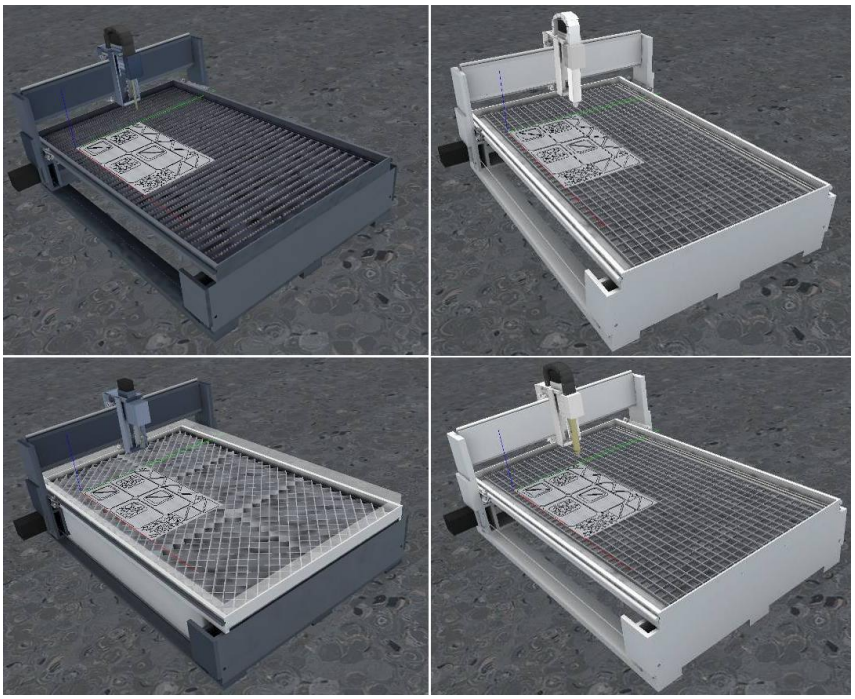

Fig 2. A specialized cutting machine model has been provided for each technology: oxy-fuel, laser, plasma and water-jet.

Finally, thanks to the multilingual support from Qt, the interface has been easily translated to several languages, with an easy and portable mechanism to add new languages.

3.3 Machine Representation

Each cutting process has its own 3D model of the cutting machine. Although all of them behave essentially in the same way, the visual differences are important for the users. In the **Fig** 2, the different machines are shown from a similar point of view.

They are composed of a stationary table, a moving bridge over the table (*X axis*) and a moving tool-support over the bridge (*Y axis*). The tool-support contains the tool, which can be replaced with different torches depending on the ongoing work.

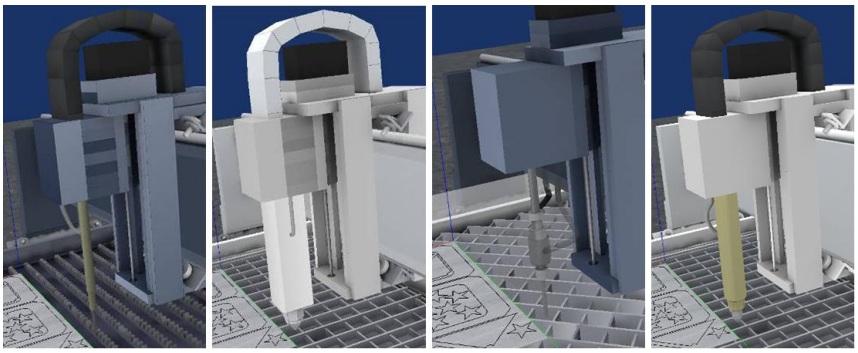

Fig 3. Each machine has its own model to represent the torches.

The bed is the most noticeable difference in the table. For example, the water jet machine provides a glass frame to avoid water splattering, and the bed's floor is solid. In the other machine, the bed's floor differs, as they are oriented to different materials and thickness of metal sheets.

Each machine has also a different torch, which provides the cutting technology. In the **Fig** 3, the torches for oxy-fuel, plasma, water-jet and laser technologies are shown.

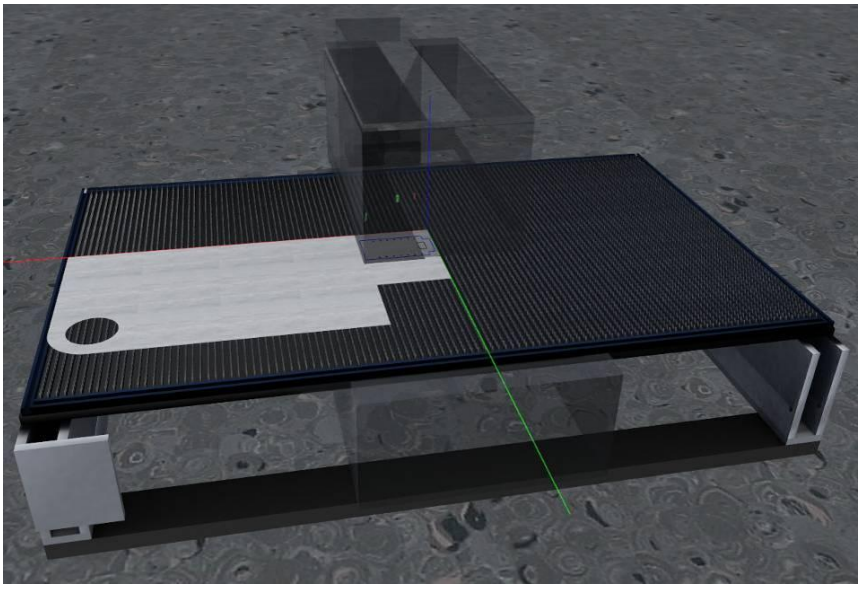

Fig 4. Punching machine 3D model. The bridge is shown with transparency to allow users to view the punching processes.

The hierarchical structure of the punching machine is significantly different from the cutting machine. In the punching machine, the sheet is the mobile ele-

ment, while in the cutting machine the sheet is static. Therefore, the sheet is moved in the X and Y axis, while the punching stations give the punch movements in the Z axis. The implemented prototype of the punching machine has room for up to 20 individual punching tools. The individual punch geometries are loaded from an XML file.

The visualization of the punching machines requires playing with transparencies in the punching station and the bridge, since they hide completely the punching operation (see **Fig** 4).

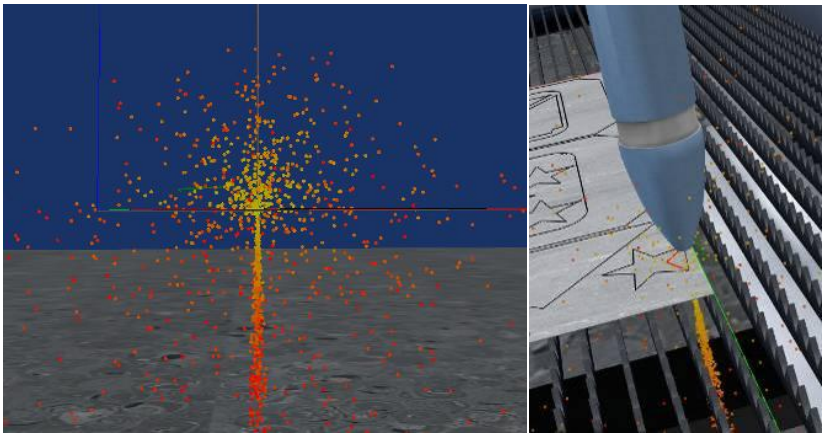

Fig 5. Visual effect to represent the sparks in the machining processes. Sparks are produced by a GLSL shader, implementing a highly configurable a particle system.

3.4 Sparks Visual Effect

The machining process produces lot of sparks when they are cutting the metal sheet. Even in punching machines, some discrete sparks can be seen when the punch is performed.

The simulation system addresses this situation by adding a visual effect for the sparks. It is basically a combination of particle effects, developed as a GPU GLSL shader (**Fig** 5). The visual effect can be configured to adapt the achieved result to different situations. For example, the colour, speed and spread angle of the particles can be adapted to match laser cutting sparks or water spread out from the torches.

4 Methodology

In the previous section the simulator features were described. The user interface and the 3D models of the machines were introduced. The missing element in the description was the definition of the metal sheet itself and how the machining operations are performed.

This section introduces the sheet representation and the cutting and punching operations support. Despite of being different processes, they are based on the removal of material from the sheet, so the internal module for such operations has been designed to be generic for such operations.

4.1 Sheet Representation

The sheet to be machined is conceptualized as a planar rectangle, defined by its contour. In fact, any given metal sheet can be defined as a set of non-intersecting and coplanar contours. Therefore, the machining operations will use 2D contour boolean operation between the sheet object and the moving tools.

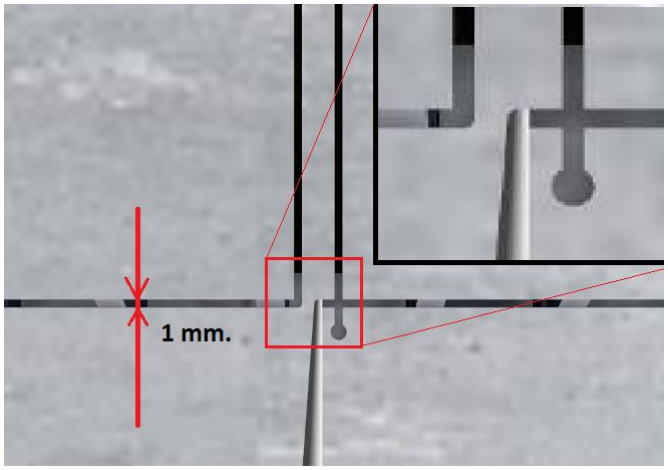

Fig 6. Cutting and punching machining. In this case, the cutting process leaves a 1mm. diameter path in the metal sheet.

As a result, the high level 3D boolean operation is simplified in a single 2D boolean operation between two 2D complex polygons, that is a well reviewed research topics [10, 7].

Normally, the metal sheet will start the machining process as a rectangular region, defined as a 4-vertices contour. As the process goes on, the path of the moving tool creates a sweep volume, which intersects the sheet. Although the torches

cut the sheet in a very narrow path (0.5 mm. or less), the swept area is noticeable (see **Fig** 6).

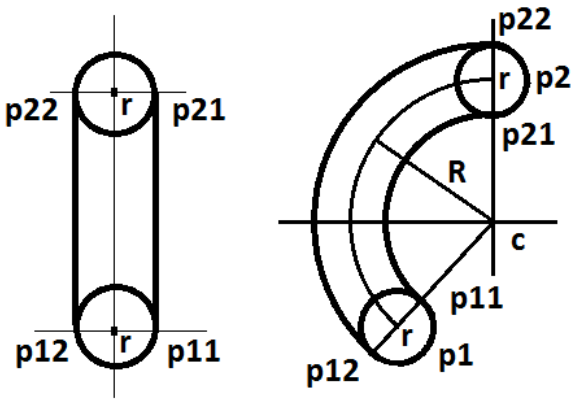

Fig 7. A moving tool produces a sweep volume. Dealing with 2D contours, the swept area can be geometrically constructed given the starting and final points; and the radiuses of the movement.

The removed material from the sheet corresponds with the swept volume of the moving tool. Taking into account that we are dealing with 2D contours and that the torch is approximated as a very thin vertical cylinder, the moving tool can be generated directly for linear and arc movements (see **Fig** 7).

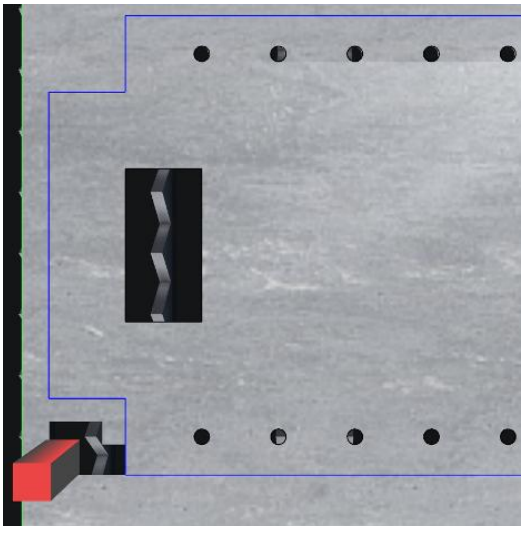

Fig 8. A punching machine produces discrete holes in the metal sheet. The different tools have different shapes to allow complex products.

The punching processes are essentially different, but the basic geometric operation remains the same (2D contour operations, see **Fig** 8). The definition of the tool profile is given as a contour, as the tool shape is not a cylinder as in the cutting processes. The swept calculation is dropped in the punching machines, as the punches are discrete, i.e., the metal sheet does not move while the punching is being performed. This situation simplifies the situation, as the calculation of a moving generic contour along a given path is not a straightforward task.

The boolean operation between the current state of the metal sheet and the swept area by the tools is performed by the clipper library [4], a polygon clipping C++ software. The result of the subtraction is the next state of the metal sheet. This iterative process continues till the end of the machining instructions.

4.2 Rendering the Geometry

The visualization of the metal sheet relies on OpenSceneGraph, the used graphics engine utilised in the simulation system. In order to render the 2D contours into the screen, a tessellation of the 2D polygons, possibly with holes, is required. This process is internally handled by OpenGL *GLUtessellator* objects and methods.

Fig 9. Render of 2D nested contours using OpenSceneGraph (GLUtesselator).

Despite of the simplicity of the approach, the results shows proper renderization of the polygons (see **Fig** 9).

The metal sheets are very thin, but they have thickness. To render the 3D dimension of the metal sheet, a duplicated geometry is rendered (top cover) and per-

pendicular planes are created to join both geometries. This skinning process is just a matter of creating a *triangle_strip* geometry per contour.

4.3 Optimization Challenges

A direct implementation of the methods proposed in this section does not give great performance to the simulation system. As the simulation goes on, the metal sheet representation grows, in number of contours, and in number of vertices. Therefore, the boolean operations become less efficient and the overall performance drops. Additionally, the rendering of very complex and nested contours is not an easy task for the underlying OpenGL subsystem, due to the tessellation process.

The next section will introduce a spatial partitioning system to improve the performance of the simulator.

5 Spatial Partition and Results

The efficiency of the polygon clipping algorithms depends directly on the total number of contours and points involved in the boolean operation [5]:

$$O (n \times \log(n) + k + z \times \log(n))$$

where n is the number of edges (points), z is the number of contours and k is the number of edge intersections. As the simulation is performed, the working part gets more and more complex and consequently, the number of points and contours grows. In order to limit the number of points and contours that would increase the boolean operation time, a high-level partitioning system is used.

Model	Movements	Subdivision	OpBools	Time (s.)
Simple	834	1×1	774	10
		4 × 4	1008	5
		16 × 16	2700	6
		32 × 32	6556	10
Complex	12731	1×1	11037	2100
		4 × 4	12481	195
		16 × 16	19533	87
		32 × 32	36910	130

Table 1 Spatial Subdivision Performance, with the direct sweep arc generation applied, varying the spatial subdivision. The number of low level boolean operation and the time are presented in the columns OpBools and Time.

This spatial partitioning decomposes the metal sheet into a set of smaller subregions, leading to a high level boolean operation pseudo-algorithm. The performance effect of the spatial subdivision is limited as an over-subdivided sheet will increase the number of individual boolean operations, as any movement will span across multiple regions. The subdivision region is aimed to reduce the complexity of the boolean operation (by limiting the number of vertices) but avoiding increasing the mean number of boolean operations per movement.

In our experiments (see Table 1), varying the number of subdivision with the same example, gives a performance peak using the 16 × 16 subdivision. The **Fig** 10 shows the geometric complexity of the metal sheet after the simulation program is completely executed. Approximately, each full circle contains around 100 vertices, 50 for the internal circle and 50 for the external circle. The whole geometric sheet contains around 1 million vertices, and due to the spatial subdivision (16 × 16 in this case), the rendering is performed in real time using the OpenSceneGraph's internal polygon tesselator (OpenGL's GLUtesselator methods).

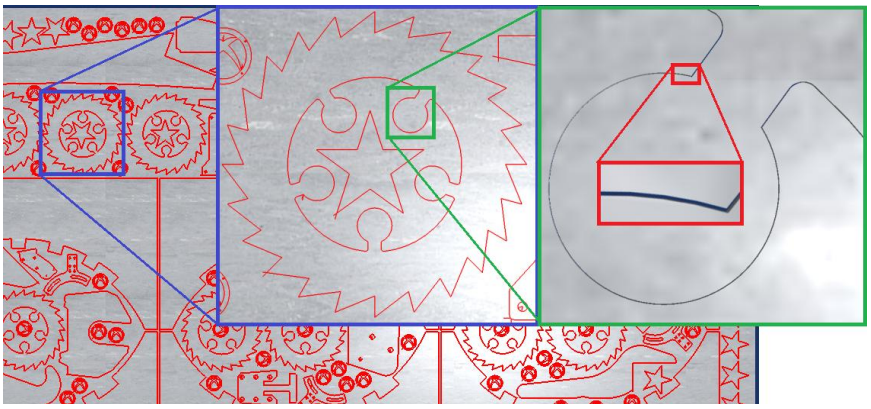

Fig 10. Simulation result of a large CNC program (12000 movements).

6 Conclusions

In this work we have introduced a representation based on 2D complex polygons to represent the metal sheet in machining processes. All the operations of the programs have been transformed to internal boolean operations between the sheet and the sweep of moving cylinder (cutting machines) or a complex polygon (for punching machines).

As the efficiency of the boolean operations decay with the number of points and polygons, we have introduced several mechanisms to optimize the overall result of the simulation. To limit the number of points and polygons, a spatial parti-

tion system has been used, checking that it cannot be increased to arbitrary numbers, since it will provoke a huge explosion in the number of boolean operations. A balanced spatial partition is required to get the best performance.

7 Future Work

Despite of the improved performance due to the utilisation of the spatial subdivision techniques, a more detailed analysis of such conditions should be evaluated. More precisely, it is not straightforward which subdivision is the best one for a given CNC program.

To address this issue, as future work, heuristic algorithm will be researched. They will analyze a given CNC program and estimate a good subdivision level which will bring a big performance increment.

Other alternative is to implement dynamic methods in the subdivision technique. As a given region becomes more and more complex, a background process could detect that situation and subdivide that region into smaller ones. The overall result in the performance should be evaluated numerically.

An extension of the proposed methodology to 3D could be addressed. A 3D object can be approximated as a set of parallel levels (see **Fig** 11**)**. Each level would contain a set of 2D contours, as the metal sheet used in this work. This extension will bring new problems into the field, such as, skinning algorithms to create a proper visualization of the 3D object, proper sweep volume calculations and conversion to a set of 2D contours and impact in the performance.

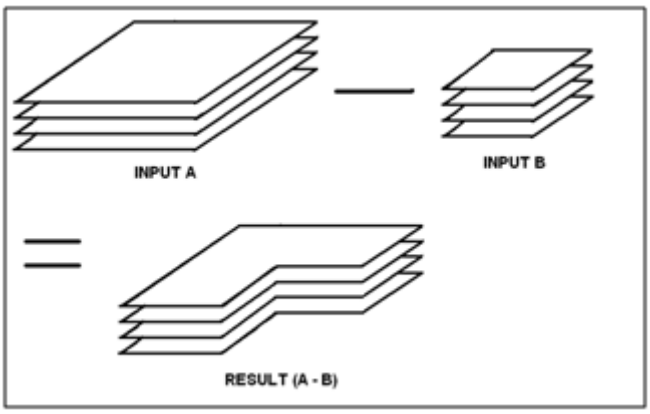

Fig 11. A boolean operation example between two 3D objects represented as a set of parallel levels.

Acknowledgments We thank the Basque Government Industry Department for the financial help received under the GAITEK research programme.

References

1. Brunet P. and Navazo I., 1990. Solid representation and operation using extended octrees. In ACM Transactions on Graphics 9, 2. 170{197.
2. Cano P., 2002. Representation of polyhedral objects using sp-octrees. In Journal of WSCG 10, 1. 95{101.
3. Hook V., 1986. Real Time shaded NC Milling Display. In SIGGraph86, Volume 20, Number 4. 15{20.
4. Johnson A., 2012. Clipper - an open source freeware polygon clipping library. URL http://www.angusj.com/delphi/clipper.php.
5. Leonov M.V., 1998. Implementation of boolean operations on sets of polygons in the plane.
6. OpenSceneGraph, 2012. Open source 3D Graphics API over OpenGL. URL http://www.openscenegraph.org/.
7. Preparata F.P. and Shamos M.I., 1985. Geometry: An Introduction. Springer-Verlag. ISBN 0-3879-6131-3.
8. Spence A.D. and Li Z., 2001. Parallel processing for 2-1/2D machining simulation. In Proceedings of the sixth ACM symposium on Solid modeling and applications. ACM, SMA '01. ISBN 1-58113-366-9, 140{148.
9. Stewart N.; Leach G.; and John S., 2003. Improved CSG Rendering using Overlap Graph Subtraction Sequences. In International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia. 47{53.
10. Vatti B.R., 1992. A Generic Solution to Polygon Clipping. Communications of the ACM, 35(7), 56{63.
11. Wallace E., 2012. Constructive solid geometry on meshes using BSP trees in JavaScript. URL http://evanw.github.com/csg.js.
12. Zhu W. and Lee Y., 2004. Product prototyping and manufacturing planning with 5-DOF haptic sculpting and dexel volume updating. In Haptic Interfaces for Virtual Environment and Teleoperator Systems. 98{105.