Title:
Simulation-Driven Method for Computing High-Speed Toolpaths for Machining Pockets

Authors:
Tathagata Chakraborty, tathagata.chakr@hcl.com, HCL Technologies Ltd.
Chinmaya Panda, chinmaya.panda@hcl.com, HCL Technologies Ltd.
Nitin Umap, nitin.umap@hcl.com, HCL Technologies Ltd.

Keywords:
Pocket Machining, High-Speed Machining, Curvilinear Toolpath, Spiral Toolpath

DOI: 10.14733/cadconfP.2023.6-10

Introduction:

Despite the popularity of 3-axis and 5-axis CNC machines, 2.5D pocket machining remains important. One can quickly remove large volumes of a given part using 2.5D roughing. There are three types of 2.5D roughing toolpaths - contour-parallel, direction-parallel, and curvilinear or spiral. Amongst these, curvilinear toolpaths are best suited for high-speed machining when capable machines and tools are available. There are, however, no simple methods for generating curvilinear toolpaths. This paper describes a quickly implementable simulation-driven method for generating curvilinear toolpaths for arbitrary pockets that uses very simple heuristics. Furthermore, our low-code approach is easier to maintain and can generate novel curvilinear toolpaths by tweaking high-level parameters and strategies. Before describing our method, we briefly review some 2.5D toolpath generation techniques.

2.5D Pocket Machining Techniques

Contour-parallel techniques have been used for machining pockets for a very long time. Here the pocket is machined by driving the tool along curves offset from the pocket boundary. The contour-parallel toolpath generally starts from a point near the center of the pocket and gradually advances toward the boundary. Alternatively, one can also use direction-parallel techniques like zig and zig-zag approaches. Here the tool is moved in a direction parallel to a given direction, either always moving in one direction (in the case of zig toolpaths) or moving alternately in opposite directions (in zig-zag toolpaths). Lastly, curvilinear or spiral toolpaths are recent developments in manufacturing. They are smoother versions of contour-parallel toolpaths and are used mainly for high-speed machining.

The toolpaths generated by contour-parallel and direction-parallel techniques are often optimized to reduce the number of retractions and the overall length of the toolpath. However, such toolpaths often require further optimization since they ignore machine dynamics [3]. Both contour-parallel toolpaths and direction-parallel zig-zag toolpaths contain many sharp corners. During machining, the tool has to slow down before it approaches a sharp corner and then speed up again as it exits the corner. The need to decelerate and accelerate repeatedly can significantly slow down the machining process. Toolpaths with sharp corners also wear down the tools faster since tool engagement increases abruptly in these corners.

High-speed machining techniques avoid and minimize sharp corners in toolpaths and maintain a near-constant tool engagement throughout the process. Smoothing corners of toolpaths generated using

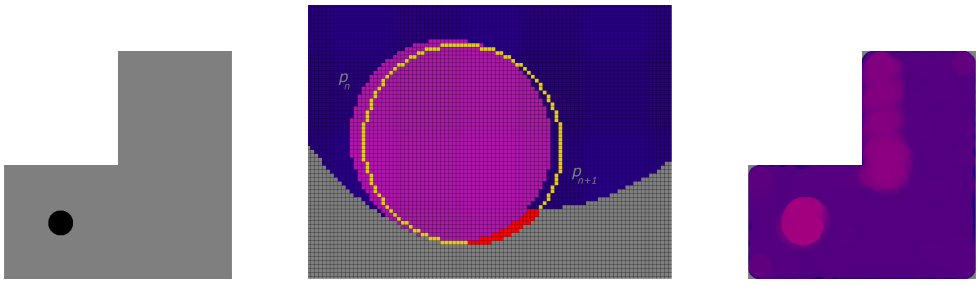

Fig. 1: Left: Simulation environment showing an L-shaped pocket with a plunge-hole. Middle: Pixel engagement computation. Right: Cutting simulation result.

contour-parallel techniques can be used to create high-speed toolpaths (see [11] [12]). In [3], a method is described where the toolpath starts as a spiral near the center of the pocket and gradually morphs into the pocket shape as it approaches the part boundary. A method based on maintaining constant tool engagement while machining the external contour of a given part is described in [5].

In [13], a method that limits the maximum curvature in the toolpath is described. In [4], curvilinear toolpath generation is framed as an optimization problem where the machining time and the forces on the tool are minimized. A continuously morphed spiral toolpath consisting of splines is proposed in [2]. A technique where the pocket is first mapped onto a circular domain and then a guide spiral in the circular domain is inversely mapped to the pocket is proposed in [14]. In [10], an analytic method for morphing a spiral toolpath to fit the shape of the pocket is presented. In [6], [7], and [1], the authors propose spiral toolpath generation methods based on the medial axis transform.

In [8] and [9], discrete and geometry-based simulation methods for analyzing and modifying toolpaths are described. In [15], a method for generating smooth toolpaths is described based on extracting iso-curves from a smoothed discrete medial axis transformed image of the pocket shape.

Pixel-based Tool Engagement Computation

Our method is based on a discrete simulation of the cutting process and is most similar to the ones described in [5] and [8], with the difference that we develop the toolpath from scratch instead of modifying an existing toolpath. The resulting toolpaths are, therefore, often better and unconstrained by the limitations of a traditional technique. A hardware-accelerated 2D rendering engine drives the simulation, using which the pocket, tool, and plunge holes are rendered on screen or onto an off-screen buffer. An illustrative screenshot is shown in Fig. 1 (left), where the pocket cross-section is colored grey, and a plunge-hole from where to start the machining process is rendered in black.

The resulting toolpath is a polyline consisting of small segments of equal length. The tool engagement over each segment is approximated by counting the difference in the number of pocket pixels when the tool is rendered at the start and end points of the segment. Such a toolpath with many small segments may not be appropriate for older CNC machines. As noted in [5], this problem can be alleviated using the Douglas-Peucker or similar algorithms. Modern NC controllers can automatically smooth toolpaths consisting of small segments.

The pixel-based tool engagement computation is illustrated in Fig. 1 (middle). Here the pocket material is in grey, the material that has already been cut is in blue, the tool at position p_n is in fuchsia, and the tool at the following position p_{n+1} is shown in outline in yellow. The additional material that gets cut when the tool moves from position p_n to p_{n+1} is highlighted in red. The count of red pixels acts as a proxy measure for tool engagement. During simulation, the tool is rendered with a graded color indicating the amount of tool engagement, enabling quick visual inspection and verification. An example

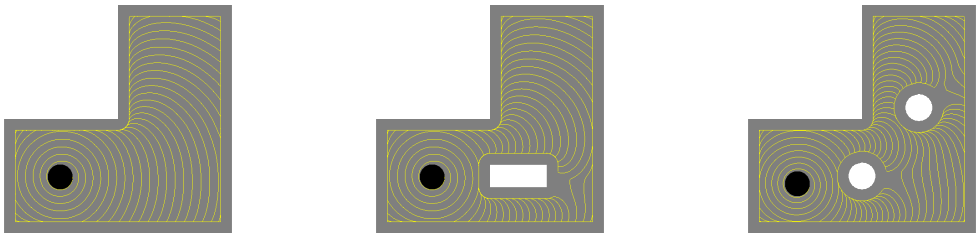

Fig. 2: Boundary following toolpaths for variations of an L-shaped pocket with islands.

is shown in Fig. 1 (right).

A Boundary Following Strategy:

A simulation-based approach allows us to develop a low-code method for computing complex toolpaths. Our strategy consists of first computing the discrete medial axis of the pocket shape. Next, a plunge hole is made at a point with the largest value in the medial axis. The tool is then placed in the plunge hole and advanced incrementally in small moves. Once the tool starts cutting material, further moves are chosen such that the tool engagement remains nearly the same, and each toolpath move deviates by only a small angle from the previous move. These two constraints are sufficient to create a spiral toolpath that stops as soon as it hits the boundary.

Algorithm 1: Overall Move Function

Input: $P_{tp} = [p_1, p_2, \cdots, p_n]$ `// toolpaths points computed till now`
Constants:
  δ `// toolpath step size`
Function `FindNextMove(`P_{tp}`):`
  $\vec{d} \leftarrow p_n - p_{n-1}$ `// direction of the last toolpath segment`
  $p_{n+1} \leftarrow p_n + \vec{d} * \delta$ `// by default move in the last direction`
  $FindBestMove(90, \vec{d}, p_{n+1}) \parallel FindBoundaryMove(90, \vec{d}, p_{n+1})$
  return p_{n+1}

It is possible to machine the complete pocket by relaxing the tool engagement constraint and making the tool move along the pocket boundary when it cannot cut any material. However, this requires evaluating a larger range of angles which can be computationally expensive. This performance penalty can be avoided by evaluating the angles in gradually expanding steps instead of evaluating the whole range. Moving the tool along the boundary in case we fail to find a cutting move can be done by including the ability to detect a gouging move. The algorithm can then select the next non-gouging angle as the direction to move when no positive tool engagement is found. A simplified version of the algorithm that computes the next move is shown in Algorithm 1, where the $FindBestMove$ tries to find the best cutting move within the specified angle range (here ± 90) and the $FindBoundaryMove$ function tries to move along the pocket boundary in case there is no valid cutting move. Both functions take the direction of the last move $\vec{d}$ as input and update the following tool location p_{n+1}.

This strategy gives us a toolpath consisting of spiral and curvilinear moves (see Fig. 2 for examples). However, it also contains a lot of redundant boundary moves. It is possible to avoid these redundant moves using a more complicated strategy, but it will be difficult to generalize such a strategy across arbitrary pocket shapes. Additionally, a more complex approach would require a large number of heuristics

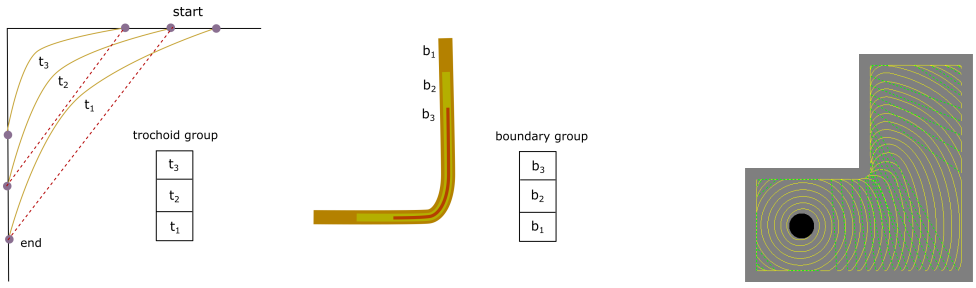

Fig. 3: Grouping of trochoidal toolpath segments (left), grouping overlapping boundary segments (middle) and the composed toolpath (right)

to handle various edge cases, making the implementation more difficult. A more straightforward and robust method for removing the redundant boundary moves and grouping the near-parallel curvilinear movements into trochoids is to use an additional post-processing step.

Post-processing the Toolpath:

In post-processing, we first identify the cutting and non-cutting sections of the toolpath. This identification is trivially made by checking the tool engagement values at each toolpath segment and grouping contiguous cutting and non-cutting segments. The next step involves identifying trochoidal groups from amongst the cutting sections. This can be done by checking if two cutting sections are nearly parallel to each other and also approximately separated by the stepover distance (see Fig. 3 (left)). Finally, we also need to identify overlapping boundary moves in the non-cutting sections discarding all but the smallest such move in each overlapping boundary group (see Fig. 3 (middle)).

Next, we connect the curvilinear moves in each trochoidal group to create the trochoidal toolpath in these regions (see the dashed lines in Fig. 3 (left)). Following this, a composition step connects the initial spiral and boundary moves with the trochoidal moves to create the complete toolpath (see Fig. 3 (right) where the dashes lines indicate linking moves). Lastly, a smoothing algorithm is used to smooth out the sharp moves in the toolpath. One can check the final toolpath for gouge and inspect the tool engagement variation by using the simulation environment to step through the tool path and create a rendering like that shown in Fig. 1 (right)

Conclusions:

Toolpath generation algorithms are usually very complex and can take much time and effort to implement robustly. The proposed method, on the other hand, requires very little code and minimal heuristics. Despite this, the method can handle complex pocket geometries and works for all pocket shapes. It is also easy to extend the technique to handle multiple plunge holes to generate more optimal toolpaths.

However, the proposed approach is still computationally more expensive than existing methods, taking several minutes, whereas geometric methods take only a few seconds on a typical desktop computer. Also, it is not easy to parallelize the method since only a single simulation environment is typically available. Therefore, it is best to deploy the method on high-end cloud machines where the results are available via a cloud-based application.

Acknowledgement:

This research was conducted as part of the CAMWorks project. CAMWorks is a popular CAM software used by small and mid-sized machining workshops all around the world. We would like to thank everyone in the CAMWorks team, past and present, who made this research possible.

References:

[1] Abrahamsen, M.: Spiral tool paths for high-speed machining of 2D pockets with or without islands, Journal of Computational Design and Engineering, 6(1), 2019, 105-117. https://doi.org/10.1016/j.jcde.2018.01.003

[2] Banerjee, A.; Feng, H. Y.; Bordatchev, E. V.: Process planning for Floor machining of $2\frac{1}{2}$D pockets based on a morphed spiral tool path pattern, Computers & Industrial Engineering, 63(4), 2012, 971-979. https://doi.org/10.1016/j.cie.2012.06.008

[3] Bieterman, M. B.; Sandstrom, D. R.: A curvilinear tool-path method for pocket machining, J. Manuf. Sci. Eng., 125(4), 2003, 709-715. https://doi.org/10.1115/1.1596579

[4] Bouard, M.; Pateloup, V.; Armand, P.: Pocketing toolpath computation using an optimization method, Computer-Aided Design, 43(9), 2011, 1099-1109. https://doi.org/10.1016/j.cad.2011.05.008

[5] Dumitrache, A.; Borangiu, T.; Dogar, A.: Automatic generation of milling toolpaths with tool engagement control for complex part geometry, IFAC Proceedings Volumes, 43(4), 2010, 252-257. https://doi.org/10.3182/20100701-2-PT-4011.00044

[6] Held, M.; Spielberger, C.: A smooth spiral tool path for high speed machining of 2D pockets, Computer-Aided Design, 41(7), 2009, 539-550. https://doi.org/10.1016/j.cad.2009.04.002

[7] Huang, N.; Lynn, R.; Kurfess, T.: Aggressive spiral toolpaths for pocket machining based on medial axis transformation, Journal of Manufacturing Science and Engineering, 139(5), 2017. https://doi.org/10.1115/1.4035720

[8] Jacso, A.; Szalay, T.; Jauregui, J. C.; Resendiz, J. R.: A discrete simulation-based algorithm for the technological investigation of 2.5 D milling operations. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 233(1), 2019, 78-90. https://doi.org/10.1177/0954406218757267

[9] Jacso, A.; Matyasi, G.; Szalay, T.: The fast constant engagement offsetting method for generating milling tool paths, The International Journal of Advanced Manufacturing Technology, 103(9), 2019, 4293-4305. https://doi.org/10.1007/s00170-019-03834-8

[10] Romero-Carrillo, P.; Torres-Jimenez, E.; Dorado, R.; Díaz-Garrido, F.: Analytic construction and analysis of spiral pocketing via linear morphing, Computer-Aided Design, 69, 2015, 1-10. https://doi.org/10.1016/j.cad.2015.07.008

[11] Stori, J. A.; Wright, P. K.: Constant engagement tool path generation for convex geometries, Journal of Manufacturing Systems, 19(3), 2000, 172-184. https://doi.org/10.1016/S0278-6125(00)80010-2

[12] Wang, H.; Stori, J. A.: A metric-based approach to 2D tool-path optimization for high-speed machining, In ASME International Mechanical Engineering Congress and Exposition, Vol. 3641, 2002, 139-148. https://doi.org/10.1115/IMECE2002-33610

[13] Xiong, Z. H.; Zhuang, C. G.; Ding, H.: Curvilinear tool path generation for pocket machining, Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 225(4), 2011, 483-495. https://doi.org/10.1177/2041297510394085

[14] Xu, J.; Sun, Y.; Zhang, X.: A mapping-based spiral cutting strategy for pocket machining, The International Journal of Advanced Manufacturing Technology, 67(9), 2013, 2489-2500. https://doi.org/10.1007/s00170-012-4666-2

[15] Xu, K.; Li, Y.; Xiang, B.: Image processing-based contour parallel tool path optimization for arbitrary pocket shape, The International Journal of Advanced Manufacturing Technology, 102(5), 2019, 1091-1105. https://doi.org/10.1007/s00170-018-3016-4