BLENDING TOOL PATHS FOR G^1-CONTINUITY IN ROBOTIC FRICTION STIR WELDING

Mikael Soron and Ivan Kalaykov
Department of Technology, Örebro University, Fakultetsgatan, Örebro, Sweden

Keywords: Robotics, friction stir welding, path planning.

Abstract: In certain robot applications, path planning has to be viewed, not only from a motion perspective, but also from a process perspective. In 3-dimensional Friction Stir Welding (FSW) a properly planned path is essential for the outcome of the process, even though different control loops compensate for various deviations. One such example is how sharp path intersection is handled, which is the emphasis in this paper. We propose a strategy based on Hermite and Bezier curves, by which G^1 continuity is obtained. The blending operation includes an optimization strategy in order to avoid high second order derivatives of the blending polynomials, yet still to cover as much as possible of the original path.

1 INTRODUCTION

Being a quite recent welding process, Friction Stir Welding (FSW), has been announced 'the next big thing' in the welding community since its release on the market a decade ago. Since its invention at The Welding Institute (TWI) in 1991 (patent filed in 92 (Thomas et al., 1991)), it has been declared a superior welding technique, in term of quality and repeatability, compared to traditional fusion welding techniques, for joining aluminium alloys.

The high quality of the FS weld is a result from a non-melting welding process without the use of filler material. And by the means of quality, it is defined through elimination (or close to) of shrinkage, porosity and distortions, which are quality issues in traditional fusion welding (Chao et al., 2003; Lomolino et al., 2005).

From an application point of view, the quality of the weld and the robustness of the process, has appealed certain types of industries with extreme demands on the results of the joining process. Two examples are the aerospace industry and in nuclear waste management (Cederqvist and Andrews, 2003), which both has an obvious zero tolerance for error. But the application does not necessarily have to be extreme to benefit from FSW, as a typical application is joining extruded aluminium profiles.

But regardless of the application area, the process is constrained by a few factors. One being the large forces and torques needed to achieve a good result. Depending on the type of material used and its thickness, a certain amount of force is applied to produce heat (by friction). These forces normally range from a few kN up to 200 kN, which calls for heavy duty machinery to support the process. And as a result from applying large forces, the object needs to be rigidly clamped in a fixture and supported by a backingbar. If the object is capable of supporting the applied forces using its internal structure, the backingbar may not be needed.

The issue of self support can in some cases be eliminated through engineering. That is, by designing the pieces properly for FSW and selecting a location of joining where it is most suited. The machinery issue, on the other hand, can not be solved with one single track. The extreme range in forces, calls for multiple solutions. A machine capable of applying 200 kN, is most certainly oversized for application in the lower region of FSW. And another issue is of course flexibility as well as the dextrous workspace of the machine. A machine capable of applying 200 kN in one direction does not necessarily have the power to apply its forces in a arbitrary direction, if even the

Soron M. and Kalaykov I. (2007).
BLENDING TOOL PATHS FOR G1-CONTINUITY IN ROBOTIC FRICTION STIR WELDING.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 92-97
DOI: 10.5220/0001624400920097

possibility to reach it, which limits the application and lowers the flexibility.

As a complement in the lower region of FSW (from a force perspective), industrial robots have been introduced. The 3-dimensional workspace of the robot as well as the relative low cost, are indeed appealing and several research project has aimed to develop a robotic solution. One, using a parallel designed robot (Strombeck et al., 2000), while the other attempts made used a serial designed robot (Smith, 2000; Soron and Kalaykov, 2006).

Even though the robots used in these applications were the ones having the greatest pay-load capabilities, the robot applications are limited in terms of material, thickness and speed due to the lack of capability to produce forces greater than approximately 10 kN. Along with the ability to apply forces, comes necessity to handle positional errors due to compliance. Both in the main direction (into the object) as well as in the plane perpendicular to the main direction.

To handle the compliance issues (especially in the main direction) force control is normally implemented in all FSW machines, robot as well as standard machines. In robotics, a standard solution is a force/position hybrid control (Raibert and Craig, 1981), enabling position based control in one or more direction, while achieving a desired force value in the other. A solution that fits FSW, since a path may be followed having a constant contact force with the material.

2 PATH GENERATION FOR FSW

Path generation has for decades been a well investigated topic for application areas such as milling (Elber and Cohen, 1993; Choy and Chan, 2003) and cutting (Wings and Jütter, 2004). To create a high defined path in such applications without an off-line programming (or CAM) tool, can be considered a time consuming operation, if not impossible in many cases.

The resemblance between mentioned processes and FSW is in many areas high, where:

- In all processes a tool is in contact with an object.
- The tool must follow a pre-defined path (or contour) with high accuracy.
- The tool's orientation must be defined at all time also often with high accuracy.

To achieve a satisfying result in such process, using an industrial robot as manipulator instead of an NC machine, a set of new motion control algorithms are essential. Typically, guidance by (3D) vision or force/torque sensing is used to compensate for the lack of positional accuracy or to gain robustness (Kim et al., 2003; Motta et al., 2001).

As mentioned earlier, force control is often used in robotic FSW applications to handle compliance issues. But also when compliance is not in play, as with traditional FSW machines, force control can increase the success rate by handling the occurrence of small deviations in the object's surface as well as indirect control the heat generation (Venable et al., 2004).

In FSW, one of the main reasons for implementing a robot application is the support of a 3-dimensional workspace. This implies that the main use of robots in FSW should be in application containing complex weld paths. At least, these are the application that benefits from a robotic solution. And as complex weld paths are in focus, so becomes the need to create those, preferably as simple as possible.

Another interesting aspect is the fact that one normally gains robustness, in force controlled motions, by having an accurate reference path. Ill-defined paths obviously need a higher amount of corrections, which the control system might be incapable of handling. And as (more) errors are introduced, new ones may arise as a result of the old.

The use of CAD objects in the path generation process is a well known technology, especially in NC applications. In robotics, on the other hand, the emphasis has been more in the area of factory modeling. The ability to visualize the robot cell (and the robot's motion), measuring cycle times as well avoiding collisions, are features supported in most of today's off-line programming (OLP) tools. But the features to model and improve a path, based on underlying CAD data and the application at hand, are often modest. Simple features, such as path blending algorithms and tool orientation control, are often hidden within more general path planning algorithms, if included at all. Even though, they are essential for many of today's robot applications, including FSW.

On most robotic systems, a path smoothing operation exists. Such operations may be defined as zones (Norrlöf, 2003) in which the control system is allowed to re-author the path in order to reach motion continuity. Such implementation are definitely helpful, e.g. when executing way-point motions, but generally not useful from a path planning perspective. Even as the FSW process is bounded to the motion of the robot, the process constraints are not accounted for in the motion controller, leaving the operator to pure guesses at the path designing stage.

3 PATH BLENDING TECHNIQUES

Blending of curves or surfaces are well known techniques when joining simple primitives to construct complex shapes. Instead of using only one equation to express a complex shape, which may lead to a high degree polynomial, it becomes natural to use several, less complex sub-shapes. And in order to maintain G^n continuity, we apply blending techniques. In terms of constructing a robot path, it becomes even more obvious since the continuity is directly linked to the ability to execute the motion. Also, in order to generate motion commands for the robot, we are restricted to use simple path primitives such as lines and arcs.

To represent a curve, with the lowest possible complexity, still offering good enough flexibility we use cubic polynomials. Expressed on a parametric form we get:

$$C(u) = (x(u), y(u), z(u)) \quad (1)$$

where $u \in [0,1]$ and

$$\begin{aligned} x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z \end{aligned} \quad (2)$$

giving the compact form as:

$$C(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d} \quad (3)$$

The tangent vector at u of the curve C is obtained by differentiating $C(u)$ as:

$$\frac{dC(u)}{du} = \left(\frac{dx(u)}{du}, \frac{dy(u)}{du}, \frac{dz(u)}{du} \right), \quad (4)$$

or as in the case with a cubic polynomial:

$$\frac{dC(u)}{du} = 3\mathbf{a}u^2 + 2\mathbf{b}u + \mathbf{c} \quad (5)$$

which are the parametric derivatives.

3.1 Hermite Blending

A cubic Hermite curve is defined based on the segments two end-points and the tangent vectors at those points. Recalling the equations 3 and 5 we obtain the following four equations:

$$\begin{aligned} C(0) &= \mathbf{d} \\ C(1) &= \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d} \\ C'(0) &= \mathbf{c} \\ C'(1) &= 3\mathbf{a} + 2\mathbf{b} + \mathbf{c} \end{aligned} \quad (6)$$

as we have substituted u for 0 and 1 respectively. Solving the equation for the four unknowns we get:

$$\begin{aligned} \mathbf{a} &= 2C(0) - 2C(1) + C'(0) + C'(1) \\ \mathbf{b} &= -3C(0) + 3C(1) - 2C'(0) - C'(1) \\ \mathbf{c} &= C'(0) \\ \mathbf{d} &= C(0) \end{aligned} \quad (7)$$

And by substituting the algebraic coefficients, we get:

$$\begin{aligned} C(u) = {} & (2u^3 - 3u^2 + 1)C(0) \\ & + (-2u^3 + 3u^2)C(1) \\ & + (u^3 - 2u^2 + u)C'(0) \\ & + (u^3 - u^2)C'(1) \end{aligned} \quad (8)$$

In a blending algorithm we may select $C(0)$ and $C(1)$ from the two intersecting curves to join them with a C^1 continuity. In such case, the only thing affecting the output of the curve is the location of those end-points. This may of course lead to unwanted phenomena in the practical case due to high second order derivatives. But if we release the C^1 continuity in order to affect the blending curve by changing the magnitude of the tangent vectors, we get another variable, k. By doing so, we still have geometric first order continuity, G^1, and a more flexible algorithm for blending. We then form the blending function as:

$$\begin{aligned} C(u) = {} & F_1(u)\mathbf{p}_0 + F_2(u)\mathbf{p}_1 \\ & + F_3(u)k\mathbf{t}_0 + F_4(u)k\mathbf{t}_1 \end{aligned} \quad (9)$$

where the F_n are the Hermite basis functions (Martens, 1990) substituted from equation 8, the $\mathbf{p}_n$ are the intersecting control points and $\mathbf{t}_n$ are the tangent vector at $\mathbf{p}_n$.

3.2 Bezier Blending

Yet another type of Spline curve, often used in blending operation, is the Bezier curve(Bezier, 1986). In difference to the Hermite curve, which interpolates all its control points, Bezier curves only interpolate the first and last. The usage of the tangents at the end-points is similar to the Hermite, but the basis functions of a Bezier curve are based on a Bernstein polynomial as:

$$C(u) = \sum_{i=0}^{n} \mathbf{p}_i \frac{n!}{i!(n-i)!} u^i (1-u)^{n-i} \quad (10)$$

for $u \in [0,1]$.

In the case of smoothing a sharp intersection, we now use three control points, giving the Bezier equation:

$$C(u) = (1-u)^2\mathbf{p}_0 + 2u(1-u)\mathbf{p}_1 + u^2\mathbf{p}_2 \quad (11)$$

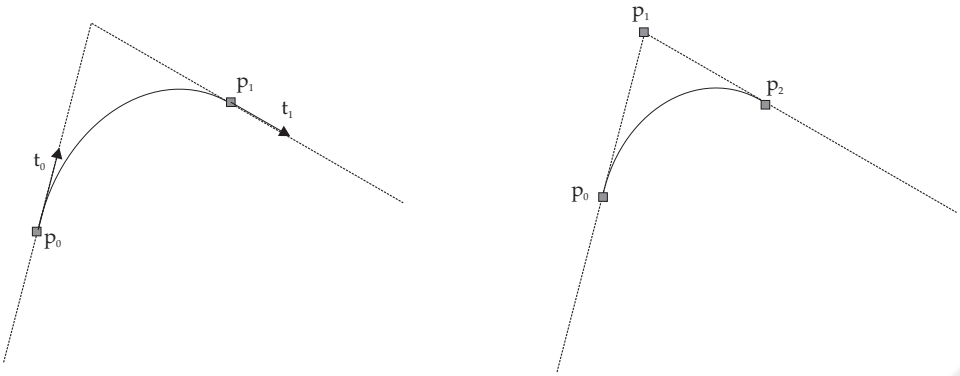

Figure 1: On the left a Hermite smoothing using two control points and its tangents. On the right a Bezier smoothing based on three control points.

As with the Hermite curve, the key to influence the behavior of the curve is through the basis functions. With Bezier curve we use the expression of weights at the control points. The weight is generalized through specifying multiple-coincident points at a vertex (control point), by which we pull the curve closer to that control point. This feature is considered desirable, due to its straight forward implementation, making easier to predict the output of the curve, than in e.g. the Hermite curve. Another important feature of the Bezier curve is that it is contained within its convex hull.

4 IMPLEMENTATION

In our path generation system we implement both Hermite blending as well as Bezier blending, to allow two different approaches towards the same goal. The objective is of course to create G^1 continuous path for the robot to execute, with only a limited interaction with operator/programmer.

In both cases we allow the user to define a minimum and maximum distance from the intersection point, where the blending function should be applied. The idea behind having an interval in which the blending is performed is to allow the system to optimize its blending performance. In the case with the Hermite curve we also allow an interval for the magnification of the tangent vector from $[0,k]$, while the Bezier is allowed to put weight on the intersecting control point, $\mathbf{p}_1$.

Since neither Hermite or Bezier curves have any corresponding motion implementation on the robot's control system, the blending motion needs to be executed as a circular motion (or rather as a sequence of consecutive linear/circular motions). Therefore, at each iteration of the blending optimization, the curve is converted into circular path segments, which endpoints tangent's are evaluated as well as the segments second order derivative. To avoid a too time consuming operation, we introduce discrete steps on which we perform the optimization.

4.1 Blending Sequence

The following sequence is used in the Hermite blending operation:

1. Set initial variables for magnification ($m = 1$), error ($\varepsilon_0 = \infty$), optimization tolerance (μ) and distance interval ($d \in [d_{min}, d_{max}]$)
2. Define the intersection to blend through original path segments, $C_0(u)$ and $C_1(u)$
3. If $d < d_{max}$ exit
4. Extract control points at $\mathbf{p}_0 = C_0(1-d)$ and $\mathbf{p}_1 = C_1(d)$
5. Calculate the tangents, $\mathbf{t}_0$ and $\mathbf{t}_1$, at $\mathbf{p}_0$ and $\mathbf{p}_1$
6. Create the blending curve, C, and calculate highest, $|C''|$.
7. If $\varepsilon_i < \varepsilon_{i-1}$ increase m jump to 5.
8. If $\varepsilon > \mu$ reset m increase d and jump to 3.
9. Exit

By applying this strategy we iterate until the tangent error is less than the optimization tolerance. If the distance interval is zero, $d_{min} = d_{max}$ we only optimize locally on the magnification factor.

When using the Bezier method we end up with a similar approach, namely:

1. Set initial variables for weight ($w = 0$), error ($\varepsilon_0 = \infty$), optimization tolerance (μ) and distance interval ($d \in [d_{min}, d_{max}]$)

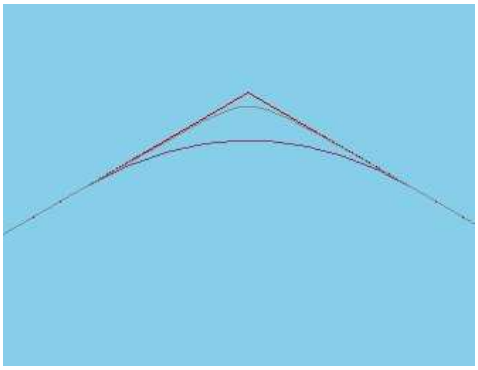

Figure 2: The resulting curves after blending.

2. Define the intersection to blend through original path segments, $C_0(u)$ and $C_1(u)$
3. If $d < d_{max}$ exit
4. Extract control points at $\mathbf{p}_0 = C_0(1-d)$, $\mathbf{p}_1 = C_1(0)$ and $\mathbf{p}_2 = C_1(d)$
5. Create the blending curve, C, and calculate highest, $|C''|$.
6. If $\varepsilon_i < \varepsilon_{i-1}$ increase m jump to 4.
7. If $\varepsilon > \mu$ reset m increase d and jump to 3.
8. Exit

In this approach we still iterate through an inner and an outer loop. But instead of evaluating the inner loop on magnification we evaluate the weight, w.

One remark should be that the Bezier curve has its convex hull property, which does not allow it to expand outside volume of the control points. This is however not the case with the Hermite, which might exhibits undesirable characteristics, such as loops and cusps (Mortenson, 2006).

5 SIMULATION

In this experiment we compare the output of our two strategies in a path modeling environment. We will not declare one better than the other, but simply visualize the output curves using the same input path segments. By using the same input parameters for the common variables, such as distance interval and tolerance level.

The two input segments are defined in the xy plane through three control points as:

$$\begin{aligned} \mathbf{p}_1 &= [0,0,0] \\ \mathbf{p}_2 &= [0,100,0] \\ \mathbf{p}_3 &= [100,100,0] \end{aligned}$$

where $\vec{s_1} = \mathbf{p}_2 - \mathbf{p}_1$ and $\vec{s_2} = \mathbf{p}_3 - \mathbf{p}_2$. The two segments $\vec{s_1}$ and $\vec{s_2}$ correspond to two parameterized curves, denoted $C_1(u)$ and $C_2(u)$ with $u \in [0,1]$.

As input parameters to our blending algorithm we choose:

- A bending distance interval ranging from $d_{min} =$ 10 to $d_{max} = 50$.
- A tangent magnification factor of $k_{max} = 5$ (for Hermite).
- A weight of $w_{max} = 5$ (for Bezier).

The resulting intersection blending of the two segments, $\vec{s_1}$ and $\vec{s_2}$, is shown in Figure 2. Reviewing the two operations, one may notice that the Bezier algorithm, due to its convex hull property, deviates less from the original path than the Hermite algorithm. The limiting factor in the Bezier case is rather the rapid increase of the second order derivative. In the case of the Hermite operation, the change of the magnification factor pulls the curve closer to (or further away from) the intersecting control point. But in this case, overexceeding the limits causes an overshot of the curve. The visual feedback in such case, is a helping factor in the path planning stage.

One may conclude the simulation with the following remarks:

- The Bezier algorithm allows less deviation from the original path, but the second order derivatives needs to be monitored. One may also mention the straight-forward solution to manipulate the curve by adding weights at control points.
- The Hermite solution have a more complex approach to manipulation, but have on the other hand a visual response when overexceeding limits.

6 CONCLUSIONS

In this paper we have suggested two methods on how to blend path segments to create G^1 continuous paths. There certainly exists numerous of way to do this, but the lack of implementations calls for such studies. In our solution we provide the following:

- Two different implementations, Bezier or Hermite blending.
- Allowing the user to define the location (or location interval) of the blending function.
- A local optimization based on vector magnitude (Hermite) or weight (Bezier).
- A global optimization on the interval (if existing).

One issue that occurs in this type of implementations is that the blending function needs to be converted into executable motion, since spline motions rarely exists in the control system of the robot. One could probably create better implementations through subdividing the splines until a certain level of accuracy is obtained, but this is, however, not within the scope of this implementation.

The idea of intersection blending enlightens only a portion of all issues involved in path generation. It is, however, an essential issue in order to achieve good weld result in 3-dimensional FSW, since the process output is depending on a continuous motion of the welding tool.

ACKNOWLEDGEMENTS

The first author's research is financed by the KK-foundation and ESAB AB Welding Equipment. Their support is highly appreciated. The first author is grateful for the technical support provided by the project members ESAB AB Welding Equipment, ABB Robotics and the research centre for Applied Autonomous Sensor System (AASS) at Örebro University.

REFERENCES

Bezier, P. (1986). *The Mathematical Basis of the UNISURF CAD System*. Butterworth-Heinemann, Newton, MA, USA.

Cederqvist, L. and Andrews, R. E. (2003). A weld that last for 100000 years:fsw of copper canisters. In *CD-ROM Proceedings of the 4th International Symposium on Friction Stir Welding, Park City, Utah*.

Chao, Y. J., Qi, X., and Tang, W. (2003). Heat transfer in friction stir welding - experimental and numerical studies. *Manufactoring Science and Engineering*, 125:138–145.

Choy, H. and Chan, K. W. (2003). Modeling cutter swept angle at cornering cut. *International Journal of CAD/CAM*, 3(1):1–12.

Elber, G. and Cohen, E. (1993). Tool path generation for freeform surface models. In *SMA '93: Proceedings of the Second Symposium on Solid Modeling and Applications*, pages 419–428.

Kim, S. I., Landers, R. G., and Ulsoy, A. G. (2003). Robust machining force control with process compensation. *Manufacturing Science and Engineering*, 125:423–430.

Lomolino, S., Tovo, R., and Santos, J. D. (2005). On the fatigue behaviour and design curves of friction stir butt-welded al alloys. *International Journal of Fatigue*, 27(3):305–316.

Martens, J. B. (1990). The hermite transform-theory. *IEEE Trans ASSP*, 38(9):1595–1606.

Mortenson, M. E. (2006). *Geometric Modeling*. Industrial Press Inc., 200 Madison Avenue, New York 10016-4078.

Motta, J. M. S. T., de Carvalhob, G. C., and McMaster, R. S. (2001). Robot calibration using a 3d vision-based measurement system with a single camera. *Robotics and Computer-Integrated Manufacturing*, 17(6):487–497.

Norrlöf, M. (2003). On path planning and optimization using splines. Technical report, Tekniska Högskolan.

Raibert, M. H. and Craig, J. J. (1981). Hybrid position/force control of manipulators. *ASME J. Dynamic Meas. Control*, (102):126–133.

Smith, C. B. (2000). Robotic friction stir welding using a standard industrial robot. *2nd Friction Stir Welding International Symposium*.

Soron, M. and Kalaykov, I. (2006). A robot prototype for friction stir welding. In *CD-ROM Proceedings of the IEEE International Conference on Robotics, Automation and Mechatronics, Bangkok, Thailand*.

Strombeck, A. V., Shilling, C., and Santos, J. F. D. (2000). Robotic friction stir welding - tool technology and applications. *2nd Friction Stir Welding International Symposium*.

Thomas, W. M., Nicholas, E. D., Needham, J. C., Church, M. G., Templesmith, P., and Dawes, C. J. (1991). International Patent Application no. PCT/GB92/02203 and GB Patent Application no. 9125978.9.

Venable, R., Colligan, K., and Knapp, A. (2004). Heat control via torque control in friction stir welding. *Nasa Tech Briefs*.

Wings, E. and Jütter, B. (2004). Generating tool paths on surfaces for a numerically controlled calotte cutting system. *Coputer-Aided Design*, 36:325–331.