Mutilayer Weld Path Planning

Maggie XU

Oct 29, 2020

1 Introduction

This document aims to introduce a method for V-shape groove mutilayer path planning (matlab code in appendix). The idea is partially based on C Yang's method (2014). This method plans the welding layers, sequence, as well as all welding points (with pose in 2d) for V-shape groove. In the algorithm, the welding bead geometry has been approximated into parallelograms or trapezoids for simplification. And the manner for welding path is planned in a side-to-center, left-to-right, bottom-to-top manner. It allows such calculation on V-groove with various thickness, assembly clearance, and groove angle.

2 Methodology

2.1 Groove definition

The groove we are dealing with are V-shape only, and it takes three parameters to describe it: thickness h, assembly clearance g, and groove angle 2β, where assembly clearance describes the distance between each half of the workpiece. In this part, we assume the center of seam lies in $(0, 0)$.

2.2 Layer cutting

The cross-sectional area of the beads is related to the wire feed rate, the welding speed, the wire diameter and the deposition coefficient, which can be expressed as

$$\mathbf{S} = \frac{a_H \pi v_1 d^2}{4v_2} \tag{1}$$

where, a_H is the deposition coefficient; v_1 is the wire feed rate; d is the wire diameter; and v_2 is the welding speed. In our case, instead of using weld parameter (unknown), we calculated the cross-sectional area of the weld bead based on the geometry

$$\mathbf{S} = (t \cdot tan(\beta) + g) \cdot t \tag{2}$$

where β is half angle of V-groove, g is the assembly clearance, and t denotes the thickness of the first layer, which needs to be specified by the user. To simplify the situation, we assume each layer has the same thickness as the first layer. The total number of layer and its corresponding length would be

$$\mathbf{K} = \left\lfloor \frac{h}{t} \right\rfloor, \quad i \in K, \quad \mathbf{L_i} = g + 2it{\cdot}tan(\beta) \tag{3}$$

and the number of weld bead in each layer is given by

$$\mathbf{N_i} = \left\lceil \frac{L_i + L_{i-1}}{L_1 + g} \right\rceil, \quad i \in K \tag{4}$$

2.3 Segmentation

To this end, we have compute the rough layers of the weld groove. Now, we we need to know the offset in x, y and the angle for each welding. The shape of each segment will be either trapezoid (when it is located at the center) or parallelogram (when is not at the center). As for the parallelogram, the height and area is fixed, we can compute its length

$$\mathbf{l} = S/t \tag{5}$$

The rule for placing the two geometries is designed as such: We will go in a bottom-to-top, side-to-center, left-to-right manner. The parallelogram will first be placed consecutively along the left side, then right side. In the end, the middle is left for the trapezoid. The pseudocode is presented below

Algorithm 1: Segmentation geometry rule

```
for i = 1 : L do
  if N_i > 1 then
    lp = left parallelogram = ⌊N_i/2⌋
    rp = right parallelogram = N_i − lp − 1
    trap = trapezoid = 1
  else
    lp = 0
    rp = 0
    trap = 1
  end if
end for
```

2.4 Geometry points identification

After segmentation, we need to find the pose of points (position and direction in 2d) as well as the sequence for welding. When the torch is in different layers, the following formula is used to calculate the offset in y direction and the value of y

$$\mathbf{\Delta y_i} = i * t, \quad i \in K, \quad \mathbf{y_i} = \mathbf{\Delta y_i} \tag{6}$$

The offset in x direction changes in each layer, for each layer, we only needs $N_i - 1$ points to define the triangle and the points are obtained by calculating the parallelogram.

Algorithm 2: x coordinate for each weld

```
for i = 1 : L do
  if N_i = 1 then
    x_ij = 0
  else if N_i > 1 then
    xl = x on the left = −L_i/2
    xr = x on the right = −xl
    for j = 1 : N_i − 1 do
      if j ∈ [1, lp] then
        x_ij = xl + j ∗ l
      else if j ∈ [N_i − lp, N_i − 1] and N_i > 2 then
        x_ij = xr − (N_i − j) ∗ l
      end if
    end for
  end if
end for
```

2.5 Weld path planning

Now we have obtained all the points that forms the boundary of segments, in the next step: geometry construction and weld path identification. We project the points to form parallelogram and trapezoid. This is achieved by drawing lines with fixed length. For each point, we want to project it to the previous layer and find its corresponding intersection point. The slope of the line will depend on its region. The projected point for left parallelogram:

$$(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y}) + (\mathbf{t} \cdot \mathbf{tan}(\beta) - \mathbf{l}, -\mathbf{t}) \tag{7}$$

Similarly, for right parallelogram we have

$$(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y}) + (-\mathbf{t} \cdot \mathbf{tan}(\beta) + \mathbf{l}, -\mathbf{t}) \tag{8}$$

The actual points for welding are identified as following

Algorithm 3: x welding points

for $i = 1 : L$ **do**
  if $N_i > 2$ **then**
    for $j = 1 : lp$ **do**
      $w_{ij} = x_{ij} + (t \cdot tan(\beta) - l, -t)$
    end for
    for $j = lp : N_i - 1$ **do**
      $w_{ij} = x_{ij} + (-t \cdot tan(\beta) + l, -t)$
    end for
    for $j = N_i$ **do**
      $w_{ij} = w_{ij,j=lp} + w_{ij,j=N_i-1}/2$
    end for
  else if $N_i = 1$ **then**
    $w_{ij} = (0, 0)$
  else if $N_i = 2$ **then**
    $w_{ij,j=1} = x_{ij,j=1} + (t \cdot tan(\beta) - l, -t)$
    $w_{ij,j=2} = w_{ij,j=1} + w_{ij,j=N_i-1}/2$
  end if
end for

In the end, we assign the orientation to the welding. For the points corresponding to a parallelogram, we use half angle of the angle at the effect point, which is $(45 + \beta/2)$ degree for left and $(90 + \beta)$ degree for right. As for the points correspond to a trapezoid, 90 degree is applied.

3 Results

The whole process is illustrated below

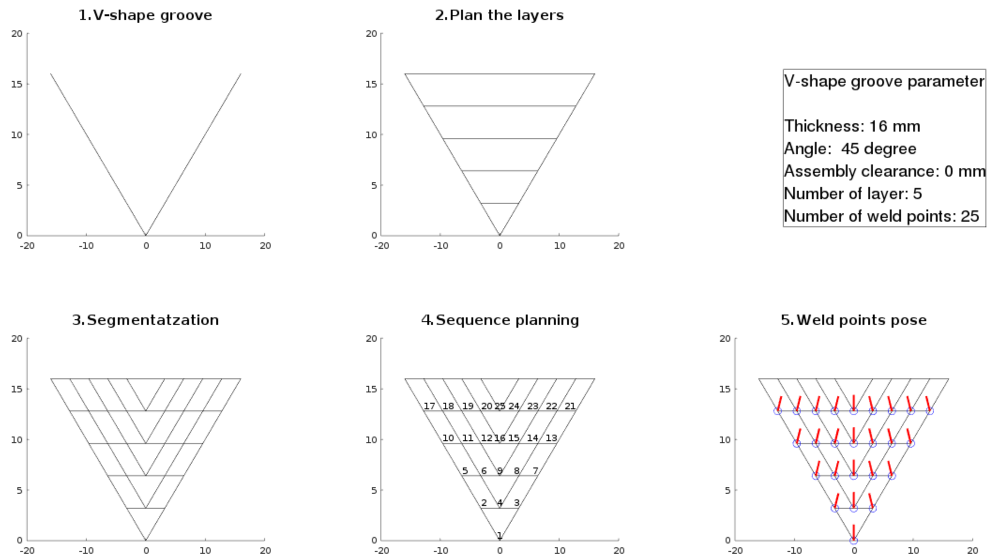

Figure 1: Process for mutilayer weld planning 1

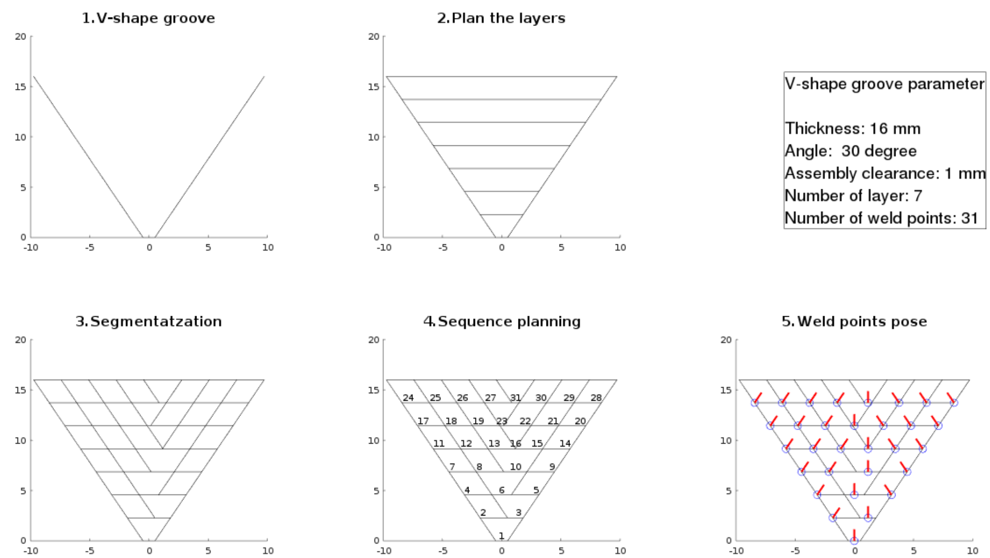

Figure 2: Process for mutilayer weld planning 2

4 Reference

Yang, C., Ye, Z., Chen, Y., Zhong, J., & Chen, S. (2014). Multi-pass path planning for thick plate by DSAW based on vision sensor. Sensor Review.

Appendix A MATLAB Code

Code is available on github: https://github.com/romi-lab/mutilayer-weld-path-planning.git