New approach to visualize Parallel Path boundaries in PragPal algorithm

Kandarp Devmurari

1 Introduction

Visualizing a new robust method to generate boundaries around the dynamic path made using PragPal algorithm

2 Previous Method

The previous method was based on a simple approach in which we kept the path width constant except at the turns. In this method the path width at turn,

$$p_{tw} = p_w / sin(\theta/2) \tag{1}$$

where $\theta = 180° - \beta$, β is the turn angle between the 2 path segment vectors.

2.1 Issue with the above method

The above method causes distortion in boundary walls when new walls are added. Since the path width is constant, the path width at the end point is p_w(as there is no more path segment for β to exist). But when a new path segment is added at this end point, the path width at this point changes from p_w to p_{tw}, because of this the previous walls distort.

3 New Method

3.1 Method-1

Using this method we can generate distortion-less parallel path boundaries but with non-constant width.

Procedure and math involved

Let us assume a 2-turn path using 4 points P_1, P_2, P_3 and P_4. Let P_1 is the start point and P_4 is the end point. Let the starting path width be p_w, $P_2 - P_1 = \overrightarrow{a}$, $P_3 - P_2 = \overrightarrow{b}$ and turn angle between $\overrightarrow{a}$ and $\overrightarrow{b}$ be β. Now we try to find the direction of average of the 2 vectors $\overrightarrow{a}$ and $\overrightarrow{b}$ to find the path boundary points at the turn.(see Figure 1)

By taking components of vector $\overrightarrow{a}$ perpendicular and parallel to $\overrightarrow{b}$ we get the average direction of the vectors. The average direction vector has a angle θ with the $\overrightarrow{b}$. Once we get the average vector direction we draw a line perpendicular to this vector at the turn point. Our boundary points will lie on this line.

By calculations from the Figure 1 and Figure 2 we get the values of θ(the angle of the average vector of $\overrightarrow{a}$ and $\overrightarrow{b}$ with $\overrightarrow{b}$), p_{tw}(path width at turn) and p_{pw}path width of path after the turn.

$$\theta = tan^{-1}\left(\frac{|a|sin(\beta)}{|b| + |a|cos(\beta)}\right) \tag{2}$$

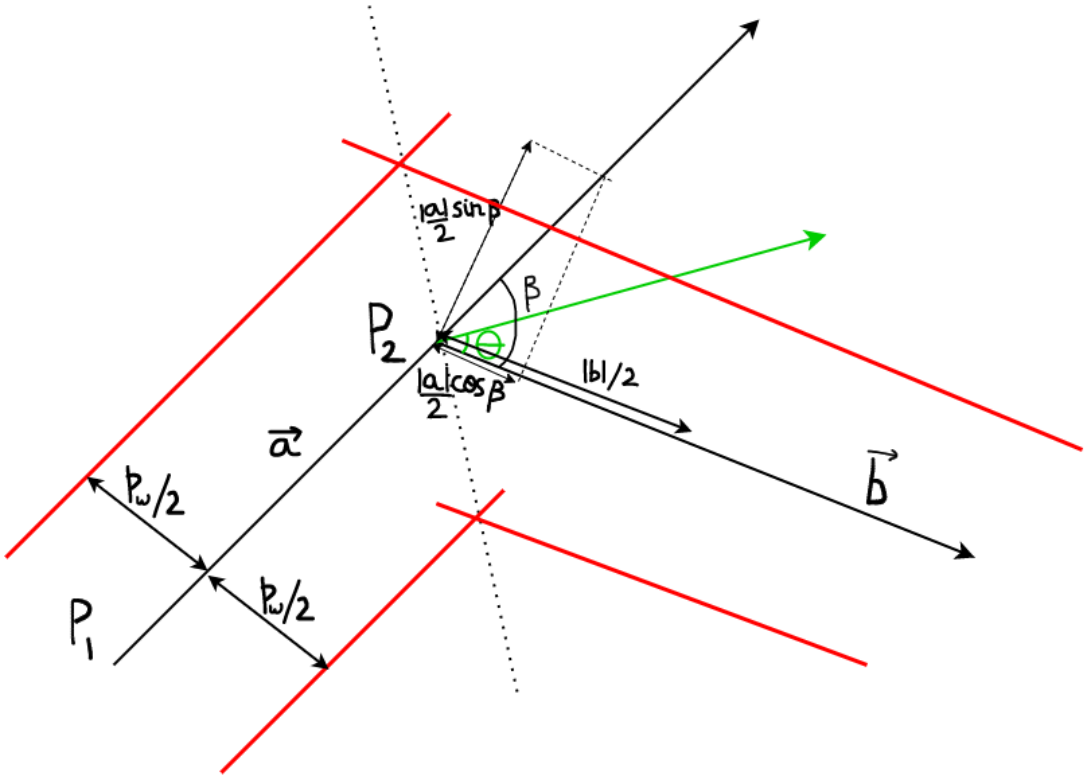

Figure 1: Finding boundary points and path width at a turn

$$p_t w = 2x = p_w \cdot sec(\beta - \theta) = p_w \cdot sec\left(\beta - tan^{-1}\left(\frac{|a|sin(\beta)}{|b| + |a|cos(\beta)}\right)\right) \tag{3}$$

$$p_p w = 2xcos(\theta) = p_t w \cdot cos(\theta) = p_w \cdot \left(\frac{cos(\theta)}{cos(\beta - \theta)}\right) = p_w \cdot \left(\frac{|b| + |a|cos(\beta)}{|a| + |b|cos(\beta)}\right) \tag{4}$$

NOTE: For $p_{pw} > p_w$, we need

$$\left(\frac{|b| + |a|cos(\beta)}{|a| + |b|cos(\beta)}\right) > 1 \tag{5}$$

$$|b| + |a|cos(\beta) > |a| + |b|cos(\beta) \tag{6}$$

$$|b| - |a| > (|b| - |a|)cos(\beta) \tag{7}$$

$$(|b| - |a|)(1 - cos(\beta)) > 0 \tag{8}$$

In the above equation, $1 - cos(\beta)$ will always be greater than 0 (unless β is 0, which implies $p_{pw} = p_w$), so for $p_{pw} > p_w$, |b| should be greater than |a| (if |b| = |a|, then again $p_{pw} = p_w$), which is always true for our algorithm. Therefore, in my method, the path width does not shrink after a turn has been made, it may increase or remain the same.

We do the same as above for all the turn points and find the line on which boundary points will lie. Then from the starting point P_1 we draw 2 parallel lines to the $\overrightarrow{a}$ and let it intersect the boundary line made previously, these intersection points will give us the left and right boundary points.

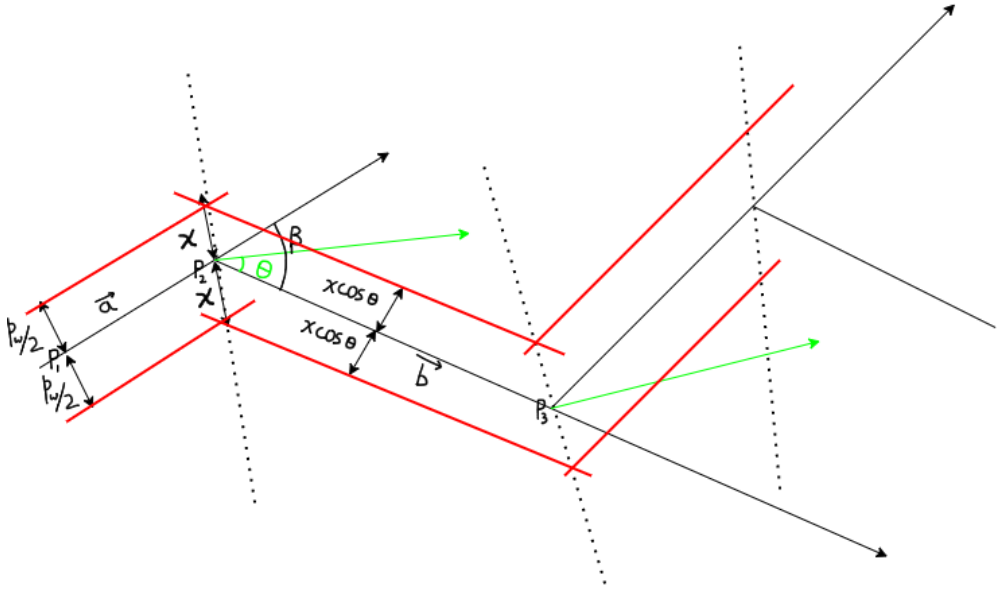

Figure 2: Generation of parallel path with non-constant path width

We repeat the above by extrapolating a parallel path again from the above newly found boundary points along the $\overrightarrow{b}$ and let it intersect the next boundary line. Using this we can generate a parallel path with non-constant width.

However, some distortion may occur at the endpoint if we are not calculating the next iterations path vector(initially the end point width was some p'_w but after a new path is added in front of it the path boundary points change making the turn path width a little longer and skewed from the original path width). This problem can be avoided if we calculate the path vector of the next segment before rendering it in VR.

3.2 Issue solved by above method

In this method if we know the (unrendered)path segment vector after the end point, so we can get boundary walls with no/minimal distortion/narrowing. Also, even if we get some distortion if will not affect the previous wall boundaries.