PATH GENERATION METHOD WITH STEERING RATE CONSTRAINT

J. Backman, T. Oksanen, and A. Visala

Department of Automation and Systems Technology
School of Electrical Engineering
Aalto University
Espoo, Finland

ABSTRACT

The practical way to generate a reference path in path tracking is to follow an adjacent swath. However, if the adjacent swath contains sharp turnings, the reference path will eventually contain sharper turn than the tractor is able to follow. This occurs especially in the corner of a field plot when the field is driven around.

In the headland, the objective is to minimize the time to reach the next swath. The commonly known method to generate the shortest path between two arbitrary positions is to use Dubins' Curves. The Dubins' Curves consist of curves with minimum turning radius and line segments connected together. The problem is that at the junction point of the path, the vehicle would have to either stop or turn wheels infinitely fast, which is not practical. Another approach is to use mathematical optimization methods to calculate the shortest path between two positions. Constraints are met, but the computational cost is high.

To overcome the problems above, a Spiral Connection method is proposed in the paper. The Spiral Connection method meets the constraints of the mechatronical steering system. If the steering rate is limited and tractor curvature is changed from the other extreme position to another, the tractor drives along a spiral. The Dubins' Curves method is modified to support spirals between curves and lines. Furthermore, in adjacent swath tracking the Spiral Connection method is used to limit the curvature of the desired path.

With the proposed method, the desired path is always feasible with respect to the constraints of the steering system. The method was implemented in an experimental guidance system, which uses Nonlinear Model Predictive Control (NMPC) to realize path tracking. The feasibility of the path is crucial to reduce the calculation time of the NMPC. In the paper, the field experiments with a tractor and a seeder are presented. The results show that the method works both in theory and in practice.

Keywords: tractors, guidance, path planning, continuous curvature, computational geometry

INTRODUCTION

The workflow of agricultural operations on the fields can be divided into four different layers: *task planning*, *path planning*, *path tracking* and *actuating*. In the first layer, *task planning*, the farmer decides when certain operations are carried out or a highly sophisticated farm management system proposes these operations. The second layer, *path planning*, is a part of an automatic navigation system and it solves the direction and the order of the driving lines in the field. The last two layers, *path tracking* and *actuating*, realizes the actual work in the field. In this research, Nonlinear Model Predictive Control (NMPC) approach is used in path tracking (Backman et al. 2012). It is previously shown that the feasibility of the target path is crucial to reduce the computational time of the NMPC and to achieve high tracking accuracy (Backman et al. 2010).

There are at least three different cases in path planning: the direction and the order of the driving lines are not limited (for example sowing); the direction of the driving lines is given but the order is free (for example silage harvesting); both the direction and the order of the driving lines are at least partially fixed (for example ploughing). The selection of the path planning method is highly dependent on the agricultural operation to be realized. Algorithms for solving the driving lines for general operation are also proposed. Oksanen and Visala (2009) have proposed two different methods for solving the direction of the driving lines: split-and-merge approach and predictive recursive online approach. The first one first splits the field into subfields which are simple to drive. After the splitting, the best driving direction for each subfield is searched. The second algorithm is an incremental algorithm that takes the machines and field current state and searches the next nearly optimal swath to be operated. Bochtis and Sørensen (2009) have proposed a method to tailor the driving order selection to commonly known vehicle routing problem (VRP). The method presumes that the optimal direction of the driving lines is predefined. Moreover, Bochtis and Sørensen (2010) have proposed a similar approach for multiple vehicles.

The direction and the order of the driving lines generally rule the path that the path tracking system should follow. However, the path is not yet feasible. The transitions between driving lines i.e. headland turnings are missing. Also, the path may contain sharp turns that should be first removed or smoothened. For the first problem, Dubins (1957) has showed that, if the car has limited curvature and only forward motion is allowed, the minimum path between two arbitrary positions is found from the set six different turning types: LRL, RLR, LSL, RSR, LSR and RSL, where “L” denotes a left turning segment with maximum curvature, “R” denotes right and “S” denotes straight segment. Furthermore, Reeds and Shepp (1990) have shown that if the backward motion is also allowed, the minimum path is found from the set of 68 different turning types consisting at most four arcs with maximum curvature and one straight line segment. However, at the junction point between different segments in the path, the curvature is discontinuous, or steps appear.

To prevent discontinuities in the Dubins’ or Reed-Shepp Curves, different solutions are proposed. Parlangeli and Indiveri (2010) have proposed a method to calculate a smooth path with bounded curvature and curvature derivative. However, the method is applicable only when there is a straight line segment between two arcs. Fraichard and Scheuer (2004) have proposed a method to extend Reeds and Shepps' turning types to paths with continuous and upper-bounded curvature and upper-bounded curvature derivative. However, in certain cases, the method does not produce paths that have optimal length. In these cases, the curvature derivative is allowed to be smaller than maximum. The method is also designed to connect only configurations with null curvature.

There are also various proposals based on numerical optimization (e.g. Fernandes et. al. 1991 and Oksanen 2007). The problem with numerical optimization is the computational complexity. The algorithms are heavy and there is no guaranteed solution at the given time window.

For the second problem, path smoothing, there are also solutions in the literature. Brezak and Petrovic (2011) have proposed a path smoothing method for smoothing a path that consists of straight line segments by using clothoids. Also Fleury et al. (1995) have introduced a method to smoothen a path described as broken lines by circles and connecting clothoids. Yang and Sukkarieh (2010) have proposed an analytical method for path smoothing by using cubic Bézier curves. Again, the original path consists of straight line segments between waypoints. In fact, all the path smoothing algorithms that the author has found are designed to smooth paths that are described as either waypoints or straight line segments.

In the presented research, it is assumed that a field plot is convex. The convex area can be covered by drawing lines with equal width side by side from the one end to another end such that none of the lines will ever go outside the field. Such field can be easily operated by driving to and fro parallel to the longest edge of the field. The remaining problem is to find feasible transitions between different driving lines i.e. headland turnings and to ensure that all the driving lines are also feasible.

METHODS

The full implementation of the path generation system is approximately 5000 lines of C++ code, so only the basic ideas are represented here. The fundamental idea is to use numerical lookup-tables for the fast evaluation of path parameters, particularly the momentary centre points of turning circles in spirals. In this manner, slow evaluations of Fresnel integrals are avoided. Also, by using numerical lookup-tables, the limit of the curvature derivative is not restricted to be constant and the resulting spiral is not necessarily Fresnel integral.

First, the idea of the connecting spirals is introduced. Then the Dubins' Curves are extended to support spirals between arcs and lines. After that, the same idea is applied to smooth a predefined path, or in other words limit the curvature of the path. Finally, a completely simplified path planning algorithm for convex field plots is presented.

Spiral Connection method

Dubins' Curves consist of six different turning types. These turnings consist of arcs with maximum curvature and straight line segment between the two arcs. At the junction point between different segments in the path, the curvature is discontinuous. To prevent these discontinuities, an extra connection segments

```
Algorithm 1. CreateSpiral
Input: $\alpha_{max}$ : maximum steering angle
       $\dot{\alpha}_{max}$ : maximum steering rate
       wheelbase : the distance between front and rear wheels
       dt : calculation resolution
Output: P($k$) : position ($x$,$y$) in a spiral corresponding to curvature $k$
        $\theta$($k$) : heading corresponding to curvature $k$
        O($k$) : centre ($x$,$y$) of turning circle corresponding to curvature $k$
        $r_1$($k_1$, $k_2$) : distance from turning circle ($k$=$k_1$) to path tangent ($k$=$k_2$)
        $r_2$($k_1$, $k_2$) : distance from turning circle ($k$=$k_1$) to path tangent ($k$=$k_2$)
```

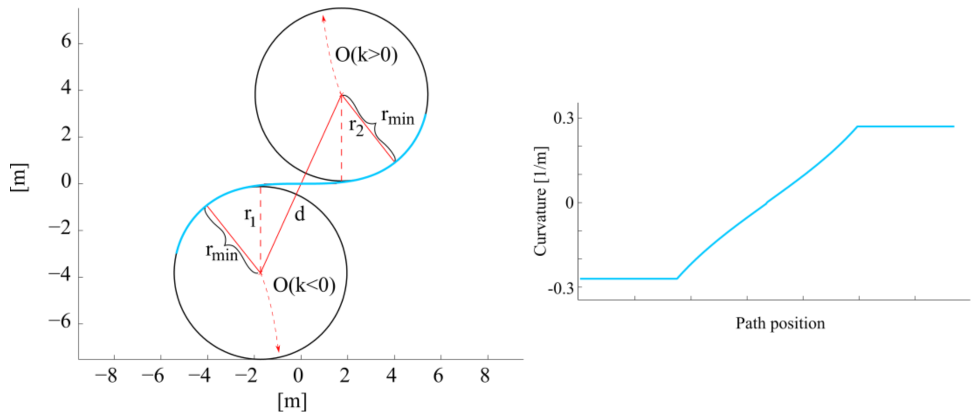


Fig 1. The car is driven with constant speed simultaneously turning wheels from the right to the left. The resulting trajectory is a spiral between two turning circles where the curvature is bounded by the properties of steering mechanics.

between every original segment is introduced. The connection segments are constructed by driving the car with constant speed and simultaneously turning the wheels from right to left with the maximum steering rate (Fig 1). The resulting trajectory represents transitions between any two momentary turning circles that the car is capable of driving. The positions and headings of the car and the centers of the turning circles are stored into a lookup table, from where the connection between two arbitrary turnings can be quickly searched. The input parameters to the spiral and lookup tables are presented in Algorithm 1.

In the following subchapters, the calculations of different turning types are presented. The LRL and RLR are the basic turning types in the headland turnings. In the first subchapter LRL turning is presented. The calculation of RLR turning is equal to LRL with the mirrored axis and for that reason omitted. In second subchapter the LSL turning is presented. The LSL turning is used if the turning distance is rather long relative to the minimum turning radius. Finally, the LSR turning is presented. The final turning path is selected from the set of feasible turnings such that the turning path has minimum length.

LRL and RLR turnings

The calculation of LRL turning is presented by pseudo code in Algorithm 2 and by drawing in Fig 2. First the starting and ending spirals are created by translating and rotating precalculated spirals such that *Spiral*$_1$ starting position equals to position *PA* and *Spiral*$_4$ end position equals to position *PB*. Then the centre of the middle turning circle is calculated by finding the crossing section of two circles, which centre points are equal to the centre points of *Spiral*$_1$ ending turning circle and *Spiral*$_4$ starting turning circle. The radii of the crossing circles are calculated from spirals (symbol *d* in Fig 1), which curvatures starts from k_{start} and end to k_{center} and from k_{center} to k_{end} equally. If such crossing section is found, the corresponding spirals between different turning circles are created again by translating and rotating precalculated spiral. Finally, the feasibility of the solution is checked. If the travelling angle within the starting or ending circle is greater than a half circle i.e. the path has a loop, the curvature of the starting or ending circle is decreased until the feasible solution is found or the decreased curvature reaches the starting or ending curvature. At last, the feasible turning path is evaluated from created spirals.

Algorithm 2. GenerateLRLTurning

```
Input: $PA$ : starting position ($x, y, \theta, k$)
       $PB$ : ending position ($x, y, \theta, k$)
Output: $Path$ : LRL turning path

while ($k_{start} > PA.k$ & $k_{end} > PB.k$)
  $Spiral_1$ = Spiral ($PA.k \rightarrow k_{start}$; $\theta(PA.k) = PA.\theta$, P($PA.k$) = $PA.\{x,y\}$)
  $Spiral_4$ = Spiral ($k_{end} \rightarrow PB.k$; $\theta(PB.k) = PB.\theta$, P($PB.k$) = $PB.\{x,y\}$)

  $O_{center}$ = CrossingOfCircles ( {$Spiral_1$.O($k_{start}$), r = |O($k_{start}$) - O($k_{center}$)| },
                                     {$Spiral_4$.O($k_{end}$), r = |O($k_{center}$) - O($k_{end}$)| })
  if( ~exists($O_{center}$) )
    return false

  $Spiral_2$ = Spiral($k_{start} \rightarrow -k_{max}$; O($k_{start}$) = $Spiral_1$.O($k_{start}$), O($-k_{max}$) = $O_{center}$ )
  $Spiral_3$ = Spiral($-k_{max} \rightarrow k_{end}$; O($-k_{max}$) = $O_{center}$ , O($k_{end}$) = $Spiral_4$.O($k_{end}$))

  if ( TurningAngleIn ($Spiral_1$.O($k_{start}$), { $Spiral_1$.P($k_{start}$) $\rightarrow$ $Spiral_2$.P($k_{start}$) } ) > $\pi$ )
    continue with $k_{start}$ = DecreateOneStep($k_{start}$)
  if ( TurningAngleIn ($Spiral_4$.O($k_{end}$), { $Spiral_3$.P($k_{end}$) $\rightarrow$ $Spiral_3$.P($k_{end}$) } ) > $\pi$ )
    continue with $k_{end}$ = DecreateOneStep($k_{end}$)

  $Path$ = EvaluatePath($Spiral_1$, $Spiral_2$, $Spiral_3$, $Spiral_4$)
end
```

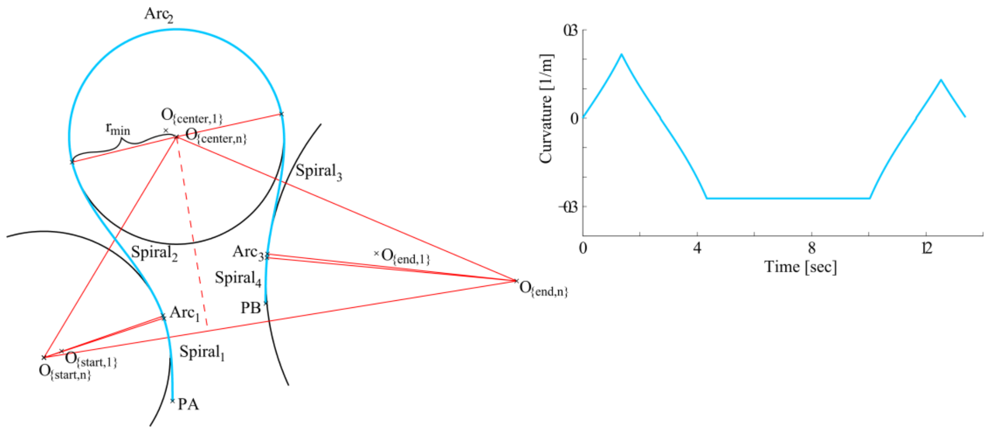


Fig 2. LRL turning from position PA to position PB using spirals consists of seven differents segments; three arcs and four connecting spirals.

LSL and RSR turnings

The calculation of LSL turning is presented by pseudo code in Algorithm 3 and by drawing in Fig 3. As in the LRL turning, first the starting and ending spirals are created by translating and rotating precalculated spirals such that *Spiral*$_1$ starting position equals to position *PA* and *Spiral*$_4$ end position equals to position *PB.* Then the orientation of the centre line is determined based on the starting and ending circles and associated distances to the path tangent (r_1 and r_2 in Fig 1). If it is possible to connect the starting and ending circles, corresponding spirals *Spiral*$_2$ and *Spiral*$_3$ are created such that spirals starting and ending circles are equal to *Spiral*$_1$ and *Spiral*$_4$ ending and starting circles respectively. Finally, the feasibility of the solution is checked with a two-part test. In the first test, it is checked that there exists a line between *Spiral*$_2$ and *Spiral*$_3$. If the spirals are intersecting, the curvature of the centre line is increased and connecting spirals to this arc is searched iteratively. If the first test is passed, the feasibility of the first and last arc is checked as in the LRL turning algorithm. If the path has a loop either in first or last arc, then the corresponding curvature is decreased and the algorithm is repeated iteratively until either starting or ending curvature is reached or feasible solution is found. At last, the feasible turning path is evaluated from created spirals.

Algorithm 3. GenerateLSLTurning

```
Input: $PA$ : starting position ($x, y, \theta, k$)
       $PB$ : ending position ($x, y, \theta, k$)
Output: $Path$ : LSL turning path

while ($k_{start} > PA.k$ & $k_{end} > PB.k$)
  $Spiral_1$ = Spiral ($PA.k \rightarrow k_{start}$; $\theta(PA.k) = PA.\theta$, $P(PA.k) = PA.\{x,y\}$)
  $Spiral_4$ = Spiral ($k_{end} \rightarrow PB.k$; $\theta(PB.k) = PB.\theta$, $P(PB.k) = PB.\{x,y\}$)
  $d = |Spiral_1.O(k_{start}) - Spiral_4.O(k_{end})|$

  while ($k_{mid} < k_{start}$ & $k_{mid} < k_{end}$)
    if ( $|r_1(k_{start}, k_{mid}) - r_2(k_{end}, k_{mid})| < d$ )
      return false

    $\varphi$ = atan2($Spiral_1.O(k_{start}) \rightarrow Spiral_4.O(k_{end})$) - asin $\left(\frac{r_2(k_{end},k_{mid}) - r_1(k_{start},k_{mid})}{d}\right)$
    $Spiral_2$ = Spiral ($k_{start} \rightarrow k_{mid}$, $\theta(k_{mid}) = \varphi$, $O(k_{start}) = Spiral_1.O(k_{start})$)
    $Spiral_3$ = Spiral ($k_{mid} \rightarrow k_{end}$, $\theta(k_{mid}) = \varphi$, $O(k_{end}) = Spiral_4.O(k_{end})$)

    if ( atan2($Spiral_2.P(k_{mid}) \rightarrow Spiral_3.P(k_{mid})$) $\neq \varphi$)
      continue with $k_{mid}$ = IncreateOneStep($k_{mid}$)
  end

  if ( TurningAngleIn ($Spiral_1.O(k_{start})$, { $Spiral_1.P(k_{start}) \rightarrow Spiral_2.P(k_{start})$ } ) > $\pi$ )
    continue with $k_{start}$ = DecreateOneStep($k_{start}$)
  if ( TurningAngleIn ($Spiral_4.O(k_{end})$, { $Spiral_3.P(k_{end}) \rightarrow Spiral_3.P(k_{end})$ } ) > $\pi$ )
    continue with $k_{end}$ = DecreateOneStep($k_{end}$)

  $Path$ = EvaluatePath($Spiral_1$, $Spiral_2$, $Spiral_3$, $Spiral_4$)
end
```

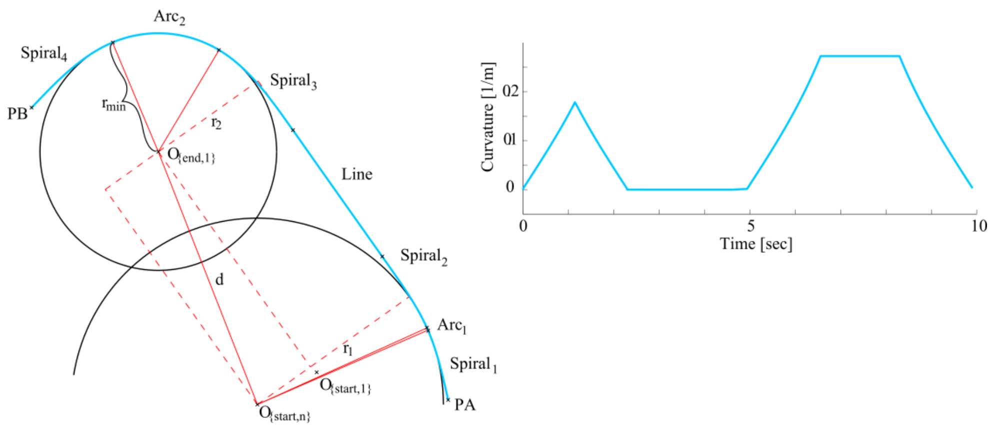


Fig 3. LSL turning from position PA to position PB using spirals consists of seven different segments; two arcs, one line and four connecting spirals.

LSR and RSL turnings

The calculation of LSR turning is presented by pseudo code in Algorithm 4 and by drawing in Fig 4. The algorithm is very similar to LSL algorithm but the curvature in the middle of the path must go through zero and for the reason the algorithm is slightly simpler. Again, the starting and ending spirals are created first. After that, the centre spirals and possible centre line is created. If the path has loops, the starting or ending circle is reduced until the path is feasible or the solution is not possible. Finally, the resulting path is evaluated from created spirals.

Algorithm 4. GenerateLSRTurning

```
Input: $PA$ : starting position ($x, y, \theta, k$)
       $PB$ : ending position ($x, y, \theta, k$)
Output: $Path$ : LSR turning path

while ($k_{start} > PA.k$ & $k_{end} > PB.k$)
  $Spiral_1$ = Spiral ($PA.k \rightarrow k_{start}$; $\theta(PA.k) = PA.\theta$, $P(PA.k) = PA.\{x,y\}$)
  $Spiral_4$ = Spiral ($k_{end} \rightarrow PB.k$; $\theta(PB.k) = PB.\theta$, $P(PB.k) = PB.\{x,y\}$)
  $d = |Spiral_1.O(k_{start}) - Spiral_4.O(k_{end})|$

  if ( $|O(k_{star}) - O(k_{end})| < d$ )
    return false

  $\varphi$ = atan2($Spiral_1.O(k_{start}) \rightarrow Spiral_4.O(k_{end})$) - asin $\left(\frac{r_2(k_{end},0)+r_1(k_{start},0)}{d}\right)$
  $Spiral_2$ = Spiral ($k_{start} \rightarrow 0$, $\theta(0) = \varphi$, $O(k_{start}) = Spiral_1.O(k_{start})$)
  $Spiral_3$ = Spiral ($0 \rightarrow k_{end}$, $\theta(0) = \varphi$, $O(k_{end}) = Spiral_4.O(k_{end})$)

  if ( TurningAngleIn ($Spiral_1.O(k_{start})$, { $Spiral_1.P(k_{start}) \rightarrow Spiral_2.P(k_{start})$ } ) > $\pi$ )
    continue with $k_{start}$ = DecreateOneStep($k_{start}$)
  if ( TurningAngleIn ($Spiral_4.O(k_{end})$, { $Spiral_3.P(k_{end}) \rightarrow Spiral_3.P(k_{end})$ } ) > $\pi$ )
    continue with $k_{end}$ = DecreateOneStep($k_{end}$)

  $Path$ = EvaluatePath($Spiral_1$, $Spiral_2$, $Spiral_3$, $Spiral_4$)
end
```

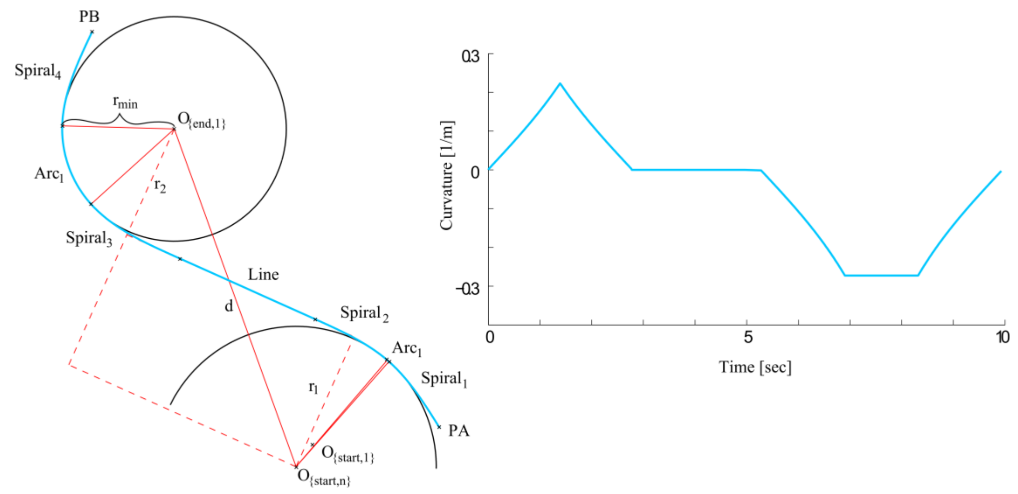


Fig 4. LSR turning from position PA to position PB using spirals consists of seven different segments; two arcs, one line and four connecting spirals.

Path smoothing

Another challenge in path generation arises when the previous driving line is used to create a new path: too sharp curves. Especially in inner curves the turning circle of the driving line decreases and eventually it is impossible to follow.

The Algorithm 5 replaces the path points in inner curves such that the resulting curve is feasible. Before the algorithm can be applied, the path is first scanned through and the starting and ending points of sharp curves are identified. The curvature limit is calculated from the working width according to equation:

$$k_{limit} = \frac{\text{sign}(follow\_dist)}{wheelbase/\tan(\alpha_{max}) + |follow\_dist|} \tag{1}$$

Algorithm 5 takes the starting and ending points of the too sharp curve and the limit values together with the original path as an input. The output of the algorithm is the smoothened path, where the path points in the neighborhood of the sharp curve are moved so that the curvature of the resulting path is less than the limit (Eq. 1).

The algorithm first creates several starting and ending spirals to the circle of maximum curvature from path points starting from given start and end points and moving further from the limited curve. The spiral starting and ending points have the same position, orientation and curvature as the original path starting and ending points have, respectively. The centre positions of the turning circles of the created spirals form two polylines. The crossing of these polylines is the centre of the desired turning circle. The starting and ending spirals to and from this turning circle can be calculated by taking weighted average of starting and ending spirals before and after the crossing section. The resulting smoothened path is obtained by moving the original path points to the nearest position in the evaluated spiral-arc-spiral path.

Algorithm 5. ReplaceSharpCurve

```
Input: $start$ : starting position in $Path$
       $end$ : ending position in $Path$
       $k_{limit}$ : maximum limited curvature
       $Path$ : original path
Output: $SmoothPath$ : new path with limited curvature

for $i$ := $start$ to $start$ - SEARCH_POINTS
    $Spiral_{start}[i]$ = Spiral ( $Path[i].k \rightarrow k_{limit}$,
                    P($Path[i].k$) = $Path[i].P$, $\theta$($Path[i].k$) = $Path[i].\theta$ )
    $O_{start}[i] = Spiral_{start}[i]$.O($k_{limit}$)
end

for $i$ := $end$ to $end$ + SEARCH_POINTS
    $Spiral_{end}[i]$ = Spiral ( $k_{limit} \rightarrow Path[i].k$,
                    P($Path[i].k$) = $Path[i].P$, $\theta$($Path[i].k$) = $Path[i].\theta$ )
    $O_{end}[i] = Spiral_{end}[i]$.O($k_{limit}$)
end

[$s$, $e$] = FindCrossing($O_{start}$[1:SEARCH_POINTS] , $O_{end}$[1:SEARCH_POINTS])

$Spiral_1$ = WeightedAvg( $Spiral_{start}$[floor($s$)] $\rightarrow Spiral_{start}$[ceil($s$)]; $weight$ = $s$-floor($s$) )
$Spiral_2$ = WeightedAvg( $Spiral_{end}$[floor($e$)] $\rightarrow Spiral_{end}$[ceil($e$)]; $weight$ = $e$-floor($e$) )

$SmoothPath$ = ReplacePathPoints( $EvaluatePath(Spiral_1, Spiral_2) \rightarrow Path$ )
```

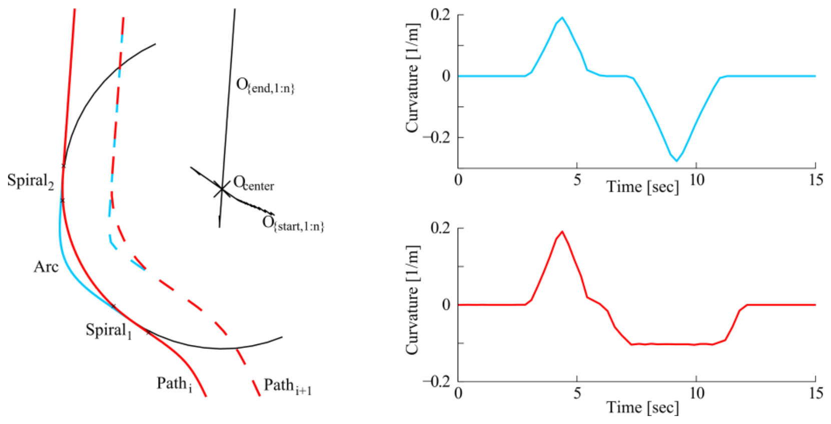
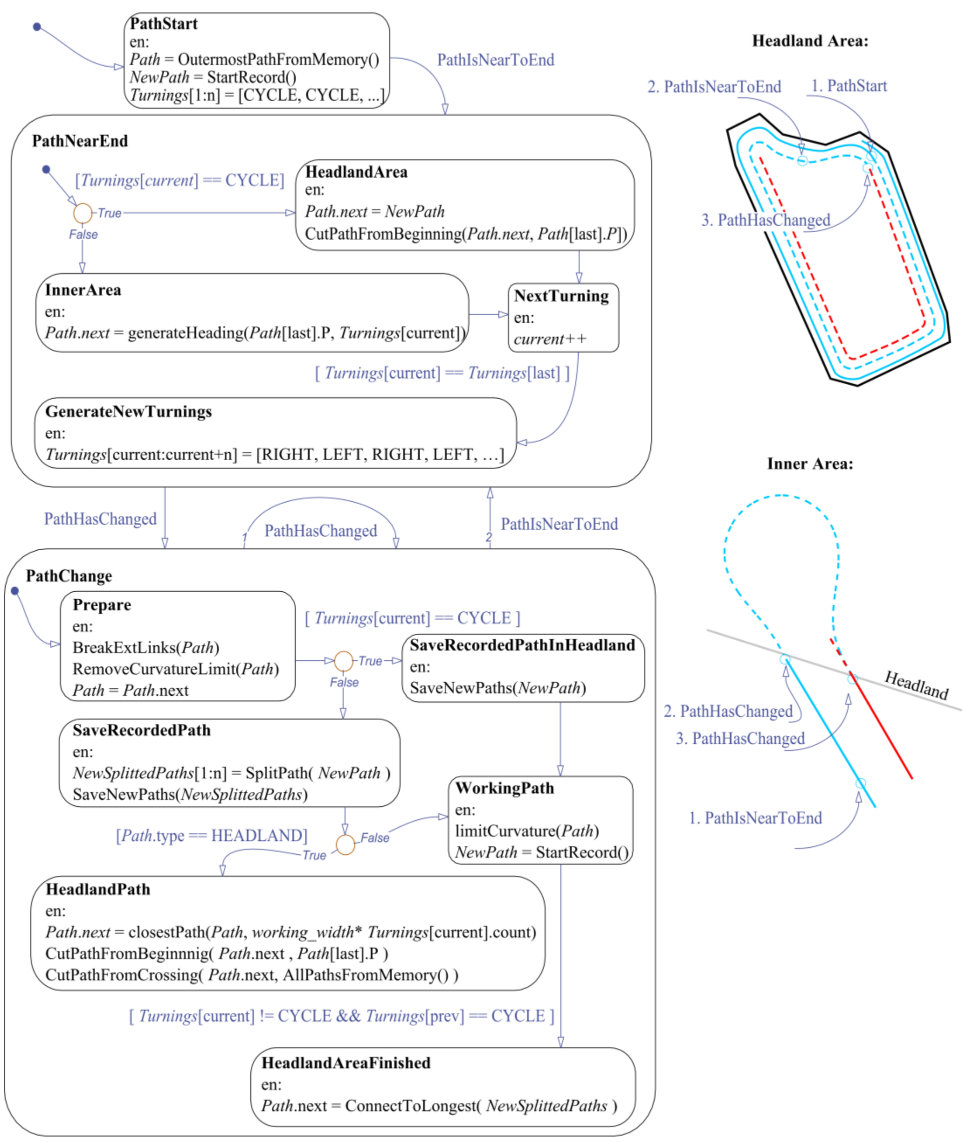


Fig 5. ReplaceSharpCurve-algorithm finds suitable spirals to connect the original (solid blue) path to the circular arch (black), which radius is equal to the minimum turning radius plus the working width. Because the gap between stored path points is relatively large, the resulting limited curvature (red plot line) is quite smooth.

Path Planning for Agricultural Vehicle

The simplified path planning algorithm for agricultural vehicles utilizes above described spiral algorithms. The basic idea of the path planning algorithm is to follow the previously driven driving lines or swaths with some offset that is multiple of working width.

The working order of the field is always the same. The field boundaries are first processed by driving around the field. After the predetermined number of cycles, the last driving line (or cycle) is decomposed into relatively straight segments according to algorithm found in Oksanen (2007) and the longest one is searched. The tracking of the cycle is continued until the longest segment is reached. After that, the processing of the inner area is continued.



PathStart
en:
Path = OutermostPathFromMemory()
NewPath = StartRecord()
Turnings[1:n] = [CYCLE, CYCLE, ...]
PathIsNearToEnd
PathNearEnd
[Turnings[current] == CYCLE]
True
False
HeadlandArea
en:
Path.next = NewPath
CutPathFromBeginning(Path.next, Path[last].P])
InnerArea
en:
Path.next = generateHeading(Path[last].P, Turnings[current])
NextTurning
en:
current++
[Turnings[current] == Turnings[last]]
GenerateNewTurnings
en:
Turnings[current:current+n] = [RIGHT, LEFT, RIGHT, LEFT, …]
PathHasChanged
PathHasChanged
1
2
PathIsNearToEnd
PathChange
Prepare
en:
BreakExtLinks(Path)
RemoveCurvatureLimit(Path)
Path = Path.next
[Turnings[current] == CYCLE]
True
False
SaveRecordedPathInHeadland
en:
SaveNewPaths(NewPath)
SaveRecordedPath
en:
NewSplittedPaths[1:n] = SplitPath(NewPath)
SaveNewPaths(NewSplittedPaths)
WorkingPath
en:
limitCurvature(Path)
NewPath = StartRecord()
[Path.type == HEADLAND]
False
True
HeadlandPath
en:
Path.next = closestPath(Path, working_width* Turnings[current].count)
CutPathFromBeginnnig(Path.next , Path[last].P)
CutPathFromCrossing(Path.next, AllPathsFromMemory())
[Turnings[current] != CYCLE && Turnings[prev] == CYCLE]
HeadlandAreaFinished
en:
Path.next = ConnectToLongest(NewSplittedPaths)
Headland Area:
2. PathIsNearToEnd
1. PathStart
3. PathHasChanged
Inner Area:
Headland
2. PathHasChanged
3. PathHasChanged
1. PathIsNearToEnd

Fig 6. The state diagram of the simplified path planning algorithm. On the right is visualized the triggering points of the state chart both in headland area work and in inner area work and the evolution of the path; the solid blue is the original path, dashed blue is the next path connected to the original and the red is the last path connected at this stage.

The state diagram of the simplified path planning algorithm is presented in the Fig 6. The algorithm has three main states: *PathStart*, *PathNearEnd* and *PathChange*. Path planning is realized when entering these states. The triggering points to these states are illustrated on the right of the Fig 6. When the headland area is processed and the path is near to an end, a new recorded path is connected to the currently followed path. When the followed path changes to this new path, recording of the new path is stopped and the path is saved to the memory. The currently followed path is also smoothened with the Algorithm 5. The process is repeated until the turning pattern has ended.

The inner area processing has the same triggering points as the headland area processing. However, when the currently followed path is a working path, the headland turning is generated before the working path has ended. When the currently followed path changes to a headland path, the closest path near to the end of the turning end point is searched and it is connected to the headland path. If there are crossing points on the connected path, those are removed i.e. the connected path is shortened such that it does not cross any previously driven path. After the headland path changes to a working path, recording is restarted. The new path is saved only when the path to be followed is a working path. The process is repeated until the whole field is processed.

RESULTS

The proposed path planning algorithm together with the Spiral Connection and the smoothing methods were implemented in an experimental automatic navigation system. The path tracking method in the experimental system is based on NMPC and the feasibility of the path is crucial for the calculation time and accuracy. First the comparison of the different turning types with the Dubins' Curves and with the Spiral Connection methods is done. Then one complete agricultural operation is reported and an extra attention is set to the case, where the curvature is limited.

In the Fig 7 are visualized different turning scenarios. The first one corresponds to normal turning to the adjacent row. The second one corresponds to turning over several rows. The last turning can be happen only if changing the row in the middle of the field or for example when changing from one subfield to another. The blue line is generated by using the Spiral Connection method and the red line is generated by the Dubins' Curves. The path with the Spiral Connection method is always longer than with maximal curvature curves, but the curvature is continuous.

The calculation time and path length were further analyzed by generating turnings between 1000 randomly chosen starting and ending points using parameters: α_{max}= 0.65, $\dot{\alpha}_{max}$= 0.4, wheelbase = 2.8 and dt = 0.1. The average calculation time of Dubins' Curves was about 2 ms and with the Spiral Connection method it was about 35 ms with non optimized Matlab-code. The average ratio between the path lengths was 1.14, meaning that headland turnings with the Spiral Connection methods were on the average 14 % longer than Dubins' Curves. In the worst case, the Spiral Connection path was 25 % longer than Dubins' Curves path.

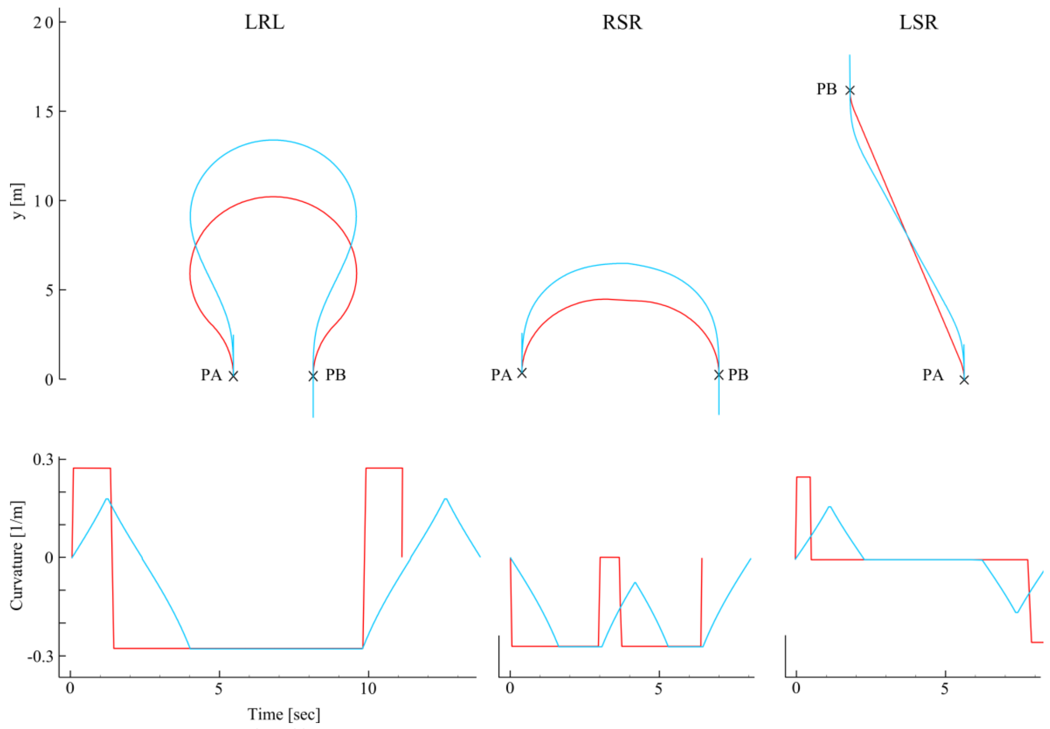


Fig 7. Comparation of different turning types: LRL, RSR and LSR.

In the Fig 8 is visualized one complete agricultural operation in the real field; seeding with towed implement. In this particular case, the field is first driven around 7 times to insure sufficient space in the headland. After that, the inner area of the field is operated by driving to and fro always turning to the adjacent swath. The automatic navigation system operated completely autonomously except when there was an electric pole in the middle of the swath at fourth cycle. At that time, the driver turned the steering wheel to avoid the obstacle. The north-west corner is enlarged to emphasize the effect of the path smoothing. Also, the turnings of the south end are enlarged to emphasize the Spiral Connection method in turnings.

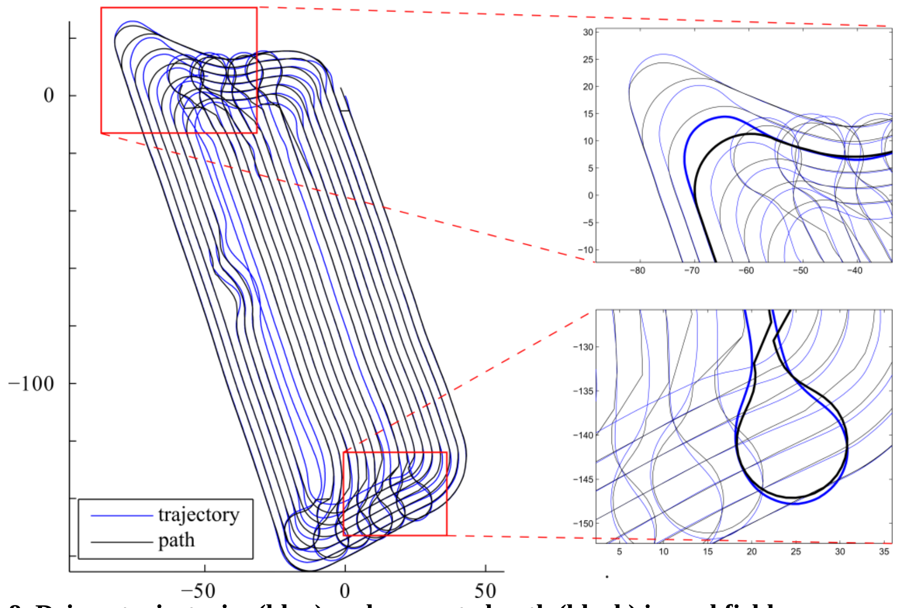


Fig 8. Driven trajectories (blue) and generated path (black) in real field.

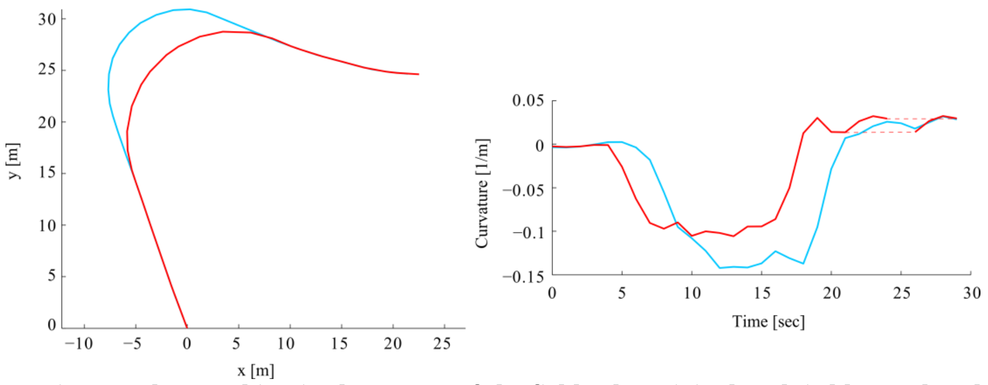


Fig 9. Path smoothing in the corner of the field. The original path is blue and path with limited curvature is red. Corresponding curvatures are in right image.

With the test equipments and with slipping present, the maximum curvature that the tractor is capable to drive is 0.14 m^{-1} meaning 23 deg steering angle with 2.8 m wheelbase. If the following distance is 3 m, the maximum curvature of the followed path is 0.10 m^{-1} according to Eq. 1. In the Fig 9 is part of the path that is showed entirely in the Fig 8. The original path curvature is -0.14 m^{-1} at most, meaning that the tractor is turning full right. The smoothened path curvature is -0.10 m^{-1} at most. In the original path, the radius of the turning circle is 7.1 m and in the smoothened path, the radius is 10 m. Therefore, if the following distance is 3 m, the radius of the turning circle remains the same about 7 m that the tractor is capable of drive.

DISCUSSION

The results show that the method works both in theory and in practice. Although the path is not analytically solved, it is still faster to calculate than with numerical optimization methods. The iteration times are bounded to be at maximum the number of half spiral elements squared in the LSL and RSR turnings that are the worst scenarios. With the parameters used in the experiments (α_{max}= 0.65, $\dot{\alpha}_{max}$= 0.4, dt = 0.1), that means 289 iterations in the worst case. The comparison of the calculation time and path length showed that the Spiral Connection method takes about 15 times longer to calculate than the Dubins' Curves, but the calculation time is still short even when the code is not optimized. The length with the Spiral Connection method is naturally longer than with Dubins' Curves, being at maximum 24 % longer in the worst case.

The steering rate is constrained by using the maximum derivative of the steering angle. Other solutions found from the literature use the maximum derivative of the curvature. The proposed solution can be modified to limit the steering rate by any function which is dependent on the current steering angle. In reality, however, the steering actuator is a dynamical system and the steering rate cannot change infinitely fast. Therefore also the second derivative of the curvature should be taken into account. However, the impact of the second derivative would be negligible.

In this paper, the objective of the path planning method was limited into convex field plots only. However, the method works in practice also for non-convex field plots. The field reported in the results is not convex, but it can be covered by this algorithm. Also, the algorithm can be extended to support most of the field types by combining it for example the split-and-merge algorithm (Oksanen et al. 2009) where the field is first divided into convex subfields. The Spiral Connection method and path smoothing can also be used separately with different path planning methods.

CONCLUSIONS

In this paper, the Spiral Connection method was proposed. With the proposed method, the desired path is always feasible with respect to the constraints of the steering system. The Spiral Connection method was applied to modify the well-known shortest path principle, Dubins' Spirals, such that the curvature of the resulting path is continuous. The method can be applied also to smoothen or constrain the curvature of an arbitrary path.

The proposed Spiral Connection method was implemented in an experimental automatic navigation system, which used NMPC as a path tracking algorithm. The results show that the method works both in theory and in practice.

ACKNOWLEDGEMENTS

A part of this research is done in the project Agromassi that is part of FIMECC-program EFFIMA. A part of this research is also funded by the Graduate School in Electronics, Telecommunications and Automation (GETA). GETA is a post graduate programme offered jointly by five universities: Aalto University School of Electrical Engineering, Tampere University of Technology and the Universities of Oulu, Turku and Jyväskylä.

REFERENCES

Backman, J., Oksanen, T., Visala, A., 2010. Nonlinear model predictive trajectory control in tractor-trailer system for parallel guidance in agricultural field operations. In: Proc. Agricontrol 2010, Kyoto, Japan

Backman, J., Oksanen, T., Visala, A., 2012. Navigation system for agricultural machines: Nonlinear Model Predictive path tracking. Computers and Electronics in Agriculture. 82, p. 32-43.

Bochtis D.D. and Sørensen C.G. 2009. The vehicle routing problem in field logistics part I. Biosystems Engineering, 104 (4), p. 447-457.

Bochtis D.D. and Sørensen C.G. 2010. The vehicle routing problem in field logistics: Part II. Biosystems Engineering, 105 (2), p. 180-188

Fernandes, C., Gurvits, L. and Li Z. L. 1991. A variational approach to optimal nonholonomic motion planning. Proceedings of the IEE Int. Conf. on Robotics and Automation. p. 680-685.

Fleury, S., Souères, P., Laumond, J. and Chatila R. 1995. Primitives for Smoothing Mobile Robot Trajectories. IEEE Trans. on Robotics and Automation, 11 (3), p. 441-447.

Fraichard, T. and Scheuer, A. 2004. From Reeds and Shepp's to Continuous-Curvature Paths. IEEE Trans. on Robotics and Automation, 20 (6), p. 1025-1035.

Dubins, L.E. 1957. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. American Journal of Mathematics, 79 (3), p. 497- 516.

Oksanen, T. and Visala, A. 2009. Coverage Path Planning Algorithms for Agricultural Field Machines. Journal of Field Robotics, 26 (8), p. 651-668.

Oksanen, T. 2007. Path Planning Algorithms for Agricultural Field Machines. Doctoral dissertation. Helsinki University of Technology.

Parlangeli, D. and Idiveri, G. 2010. Dubins inspired 2D smooth paths with bounded curvature and curvature derivative. Proceedings of the 7th IFAC Symposium on Intelligent Autonomous Vehicles 2010. p. 216-221

Reeds, J.A. and Shepp, L.A. 1990. Optimal paths for a car that goes both forwards and backwards. Pacific Journal of Mathematics, 145 (2), p. 367-393.

Yang, K. and Sukkarieh, S. 2010. An Analytical Continuous-Curvature Path-Smoothing Algorithm. IEEE Trans. on Robotics, 26 (3), p. 561-568.