Introduction to the Levee/River Calculation Methodology

Tuesday, August 20, 2013
10:53 AM

The ***Levee/River Calculation (LM Calc)*** engine is designed to locate a match point along a spatial search path (e.g. a levee or river segment) and calculate the *Marker Distance* and *Match Distance* or alternatively calculate the coordinates of a point for a specified *Marker Distance* along a designated spatial search path.

The term *Marker Distance* is the generic term used in this document instead of the more specific Levee Mile or River Mile terms. The Marker Distance calculation methodology was developed as a generic approach for calculating the Marker Distance of a given point along the levee or river starting at a defined end of the levee or river.

The levee or river is typically defined by a polyline which is as the spatial *Search Path* to locate the match point on the levee or river closest to the specified location. See Search Path Geometry Overview for an overview of typical components of the search path geometry.

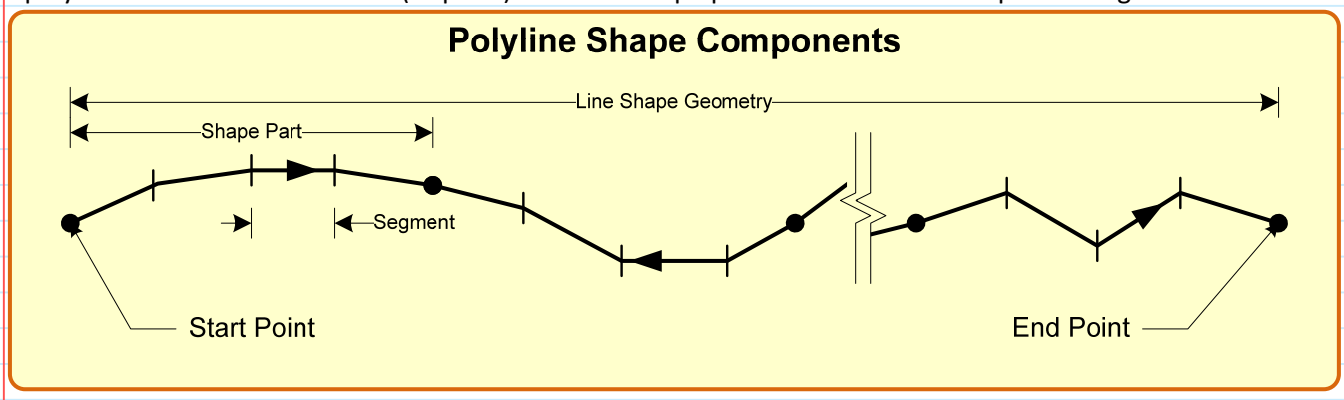
For calculation purposes, the polyline search path is represented by an object model that is referred to as a Part Tree. The Part Tree model and its subcomponents are reviewed in the section "LM Calc Components". Although the polyline search path is used in locating the closest match point, the Marker Distances along this path could be defined by a Distance Marker Overlay as discussed in section LM Calc Configuration with Distance Markers.

The section LM Calc Search Optimization reviews how the search buffer can help to optimize the spatial search process. The section on Finding Match Points of A Specified Location review the detailed calculation methodology to locate the match point for a specified location along the search path, and how the match point's Marker Distance is calculated.

Search Path Geometry Overview

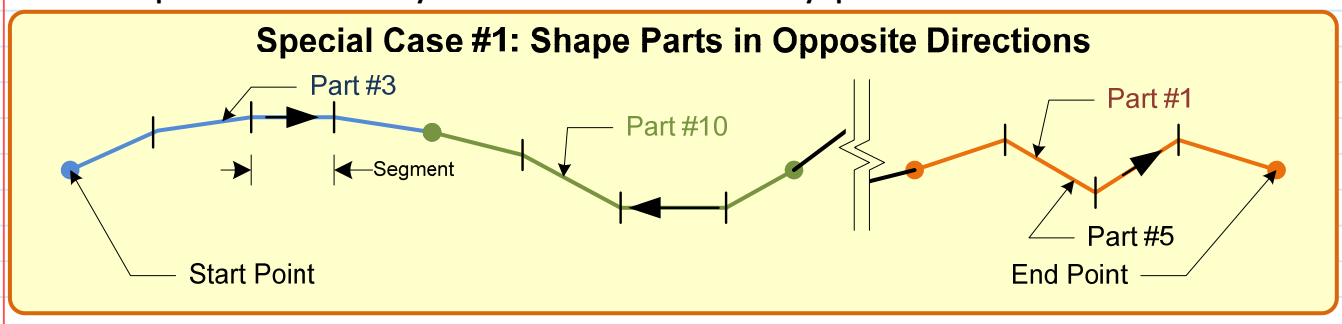
Wednesday, March 20, 2013
2:35 PM

The Search Path geometry of a levee or river is defined as a polyline shape. By definition, and as illustrated below, a polyline consist of a list of lines (or parts) and each shape part can consist of multiple line segments.



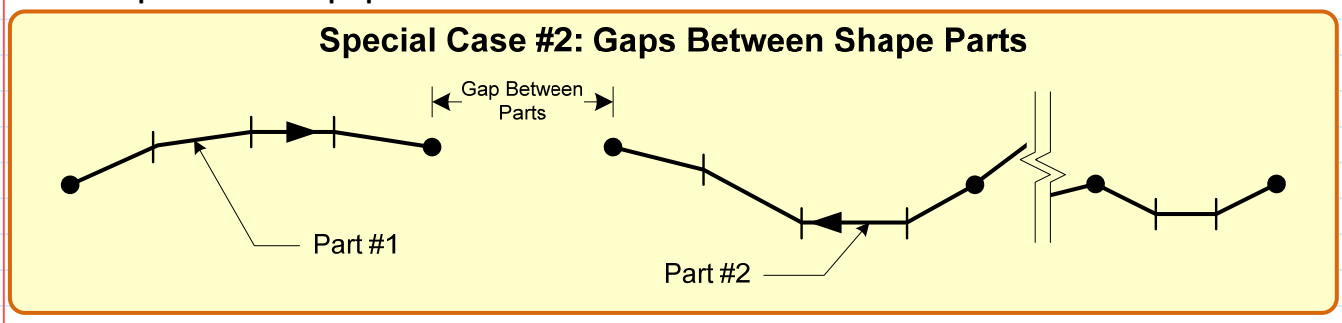
Although one generally assumes that a shape has a Start and End Point, there are several properties of a polyline (especially as used to define levee configurations) that complicates the calculation of a linear Search Distance. The following are typical cases to consider.

Case#1 : Shape Parts are not always in the same direction or in any specific order:



As shown in the above figure, there are no rules that require shape parts to be in the same direction from *Start* to *End Point*. It is also true that the shape parts can be in any random order, which implies that they are not always in the logical display order. Thus, for Marker Distance calculations it is important to identify the Distance Marker's *Start Point* and to reorganize and flip the shape parts as needed to ensure all parts are aligned in the same direction and in sequential display order.

Case#2: Gaps between shape parts:

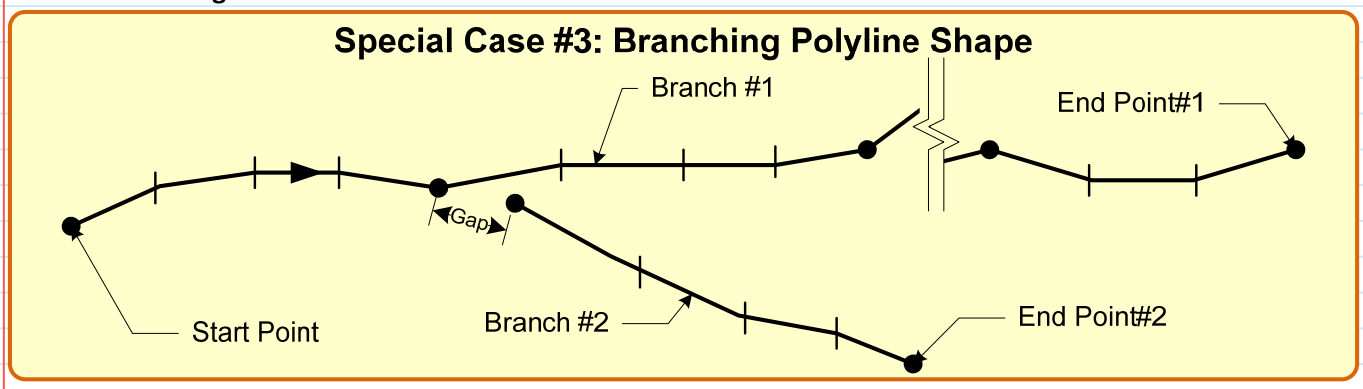


As shown in the above figure, each shape part in a polyline is an independent line with multiple line segments. This implies that there could be gaps between two neighboring parts' *End and Start Points*. In calculating the Marker Distance, these gaps are added to the Marker Distance in some cases and in other cases the gaps are ignored.

Search Path Geometry Overview (continued)

Wednesday, March 20, 2013
2:54 PM

Case#3: Branching Search Paths:



As illustrated in the above figure, a single polyline shape could include branching shape parts and the branches do not have to be closely connected. There can be gaps between the connecting points.

In the Marker Distance calculations, this is one of the more difficult cases to handle correctly - especially considering the potential variations in shape part configurations. The option used in the Distance Marker calculation approach is to use a "tree" structure to define and organize the polyline's sub-components (parts). It also implies that a polyline can have multiple End Points - one at the end of each "tree" branch.

Currently, it is assumed that the Marker Distance is always calculated as the distance from the identified *Start* Point to the point on any branch of the levee. This implies that there could be points on Branch #1 and Branch #2 with the same calculated Marker Distance.

<u>How to Prevent Duplicate Marker Distances:</u>

a) Define a Calculation Convention

In order to calculate a unique Marker Distance for each point on the search path a calculation convention should be defined that specifies how to handle these cases. For example: If Branch#2 is considered a "Sub-Branch" and Branch#1 the "Main Branch", then the convention could state that the length of Sub-Branches should be added to that of the Main Branch. Thus, the Marker Distance to any point on Branch#1 will be calculated as the distance along the polyline from the Search Path's Start Point plus the length of Branch#2.

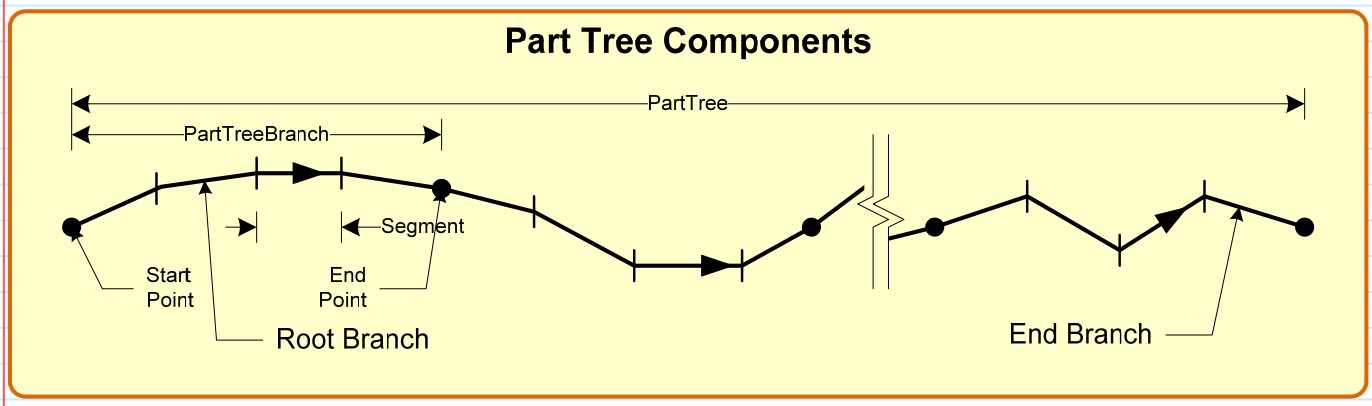
b) Split Branches into Separate Segments

The simplest, and likely most intuitive, solution is to prevent branches in the Search Path geometry by splitting all branches into new polyline shapes each with a new unique Search Path identifier.

LM Calc Components

Wednesday, March 20, 2013
3:20 PM

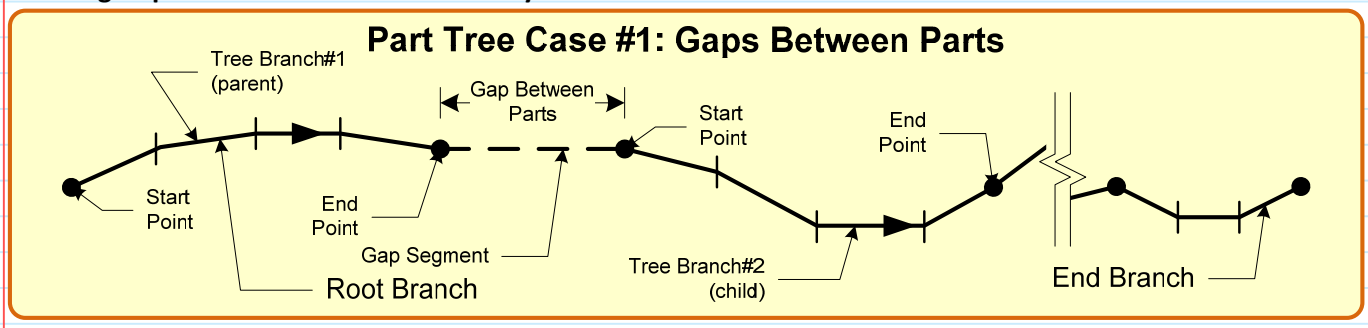
The object model used to define the Search Path geometry for the Marker Distance calculation process consists of a PartTree object (which encapsulates the polyline shape), a PartTreeBranch object (which encapsulates a polyline shape part), and a PartTreeSegment (which encapsulates a shape part's line segments). See the figure below:



Each Part Tree has a Root Branch (i.e. the first PartTreeBranch in the tree) and one or more End Branches - one at each tree branch or fork. The PartTreeBranches are configured with all shape parts pointing in the same direction from the *Start* to the *End* Point.

NOTE: In order to control the Part Tree Start-to-End configuration, the Tree's *Start* Point can be specified. Typically, this is defined as the ***Distance Marker***, which also set the PartTree's *Start* Distance (i.e., a start point offset used in calculating the Marker Distance).

Handling Gaps in the Search Path Geometry:



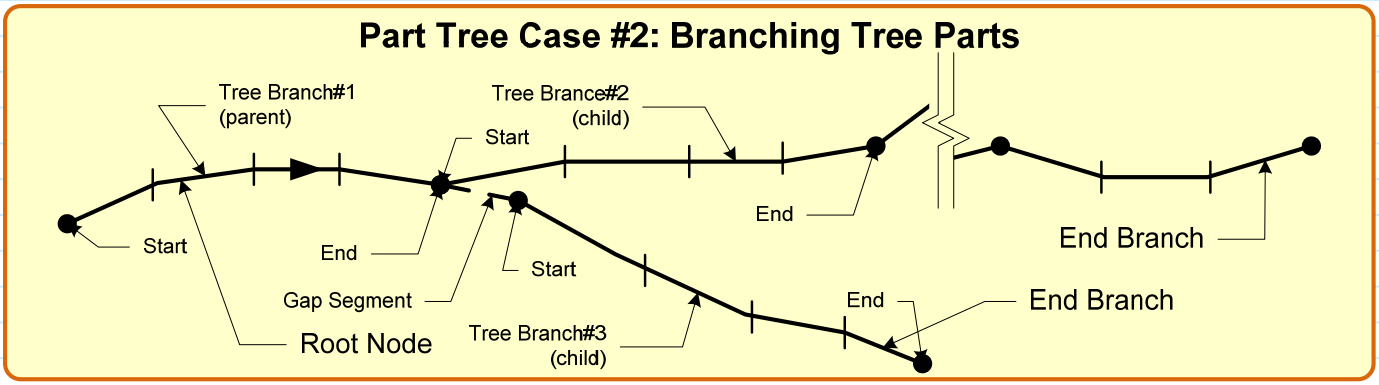
As illustrated in the above figure, if the Distance Marker calculation has to compensate for gaps between two adjacent PartTreeBranches, a Gap Segment is added to the "child" PartTreeBranch (i.e., PartTreeBranch#2 in the figure). This segment connects the end and start points of the two neighboring PartTreeBranches.

NOTE: Whether the gaps should be considered is controlled by an optional setting (*SkipGaps*) which is assigned as a PartTree property. The assumed default is to include the gaps in the Marker Distance calculations.

LM Calc Components (continued)

Wednesday, March 20, 2013
3:40 PM

Handling Branching Search Paths:



The figure above illustrates the object configuration for handling a branch/fork in the Search Path. As before all PartTreeBranches are oriented to point in the same direction and in order in a parent-child relationship starting at the PartTree's Root Branch and ending at one (or more) End Branch(es).

If there is a gap between a branch's *Start* Point and its parent PartTreeBranch's *End* Point, a Gap Segment will be used to span the gap (if the gap should be considered in calculating the Marker Distance).

LM Calc Configuration with Distance Markers

Friday, July 26, 2013
11:26 AM

Introduction

The original version of the LM Calc methodology was built on the basic approach that the Marker Distance calculations use only polyline shapes to represent the Search Path geometry. Marker Distances were based on the calculated shape length and not "actual" distances recorded using Distance (or Mile) Markers.

This version of the methodology can be used for Levee or River Mile calculations and allow the overlay of Distance (Mile) Markers to set the "actual" distance along the search path relative to a start location. This new version still supports the tree search path approach but it supports the following three options for defining the tree structure and search path distances (i.e., the Marker Distances):

1. **Using Line Shape Geometry Only:** In this case the tree search path is defined by a polyline shape and the search path distances are equivalent to the calculated shape lengths.

2. **Using Distance Markers Only**: In this case distance markers (a list of points with defined search distances (e.g., Mile Maker distance)) are used to define the search path geometry. The markers' defined locations (coordinates) are used to define the line geometry required to calculate the search match points, but instead of the calculated shape distances, the distance marker's "actual" search distances are used.

   In this case each line segment is defined as the straight line between two Distance Markers and a *MarkerRatio* is calculated as the ratio between the defined "actual" distance between the two Markers and the calculated distance between the two locations (i.e., the line segment length). This *MarkerRatio* is used in calculating the Marker Distances during searches.

3. **Using Line Shape Geometry and Distance Markers:** In this case, the Distance Markers define the "actual" distance along the search path and the shape geometry defines the spatial search path between Distance Markers. As in the *Distance Markers Only* option each line segment will have an assigned *MarkerRatio*, which is the ratio of the defined "actual" distance between two markers and the shape length between the two markers.

In order to accommodate the above input scenarios and accurately calculate these search distances and to speedup the search process, the input data include two processes:

1. First, the polyline shape data is loaded and a Part Tree is built using the line shapes only (not applicable to scenario 2 above). This process will organize each polyline shape's parts in the correct start-to-end orientation, add gap Tree Segments as needed, etc.

2. Secondly, using the points of the polyline shape's Part Tree structure and the assigned Distance Markers, a new Part Tree that includes all points is built. In this process, each Tree Segment (the line between adjacent points) is assigned a *MarkerRatio*, which is equal to the ratio between the "actual" distance between two adjacent Distance Markers and the length of the new geometry path between the two points (see Figure 1 below).

LM Calc Configuration with Distance Markers (continued)

Friday, August 16, 2013
8:47 AM

In this process, the following rules were applied to the re-building of the new Part Tree geometry:

a. If Distance Markers are used, the line geometry will include vertices at the match point (i.e. the closest point on the line perpendicular to the Distance Marker's location) of these markers. The Distance Marker's assigned "actual" distance is assumed to be the "actual" distance at this match point.

b. The Tree Branches can be limited to a *Maximum Branch Length*. In rebuilding the geometry shapes, any branch lengths will be limited to this maximum length (see Search Optimization discussion below).

c. In rebuilding, gaps in the original geometry will be spanned (filled in) if the gap is included in the Marker Distance calculation. Otherwise, the new geometry will maintain the gap and the search process will skip the gap.

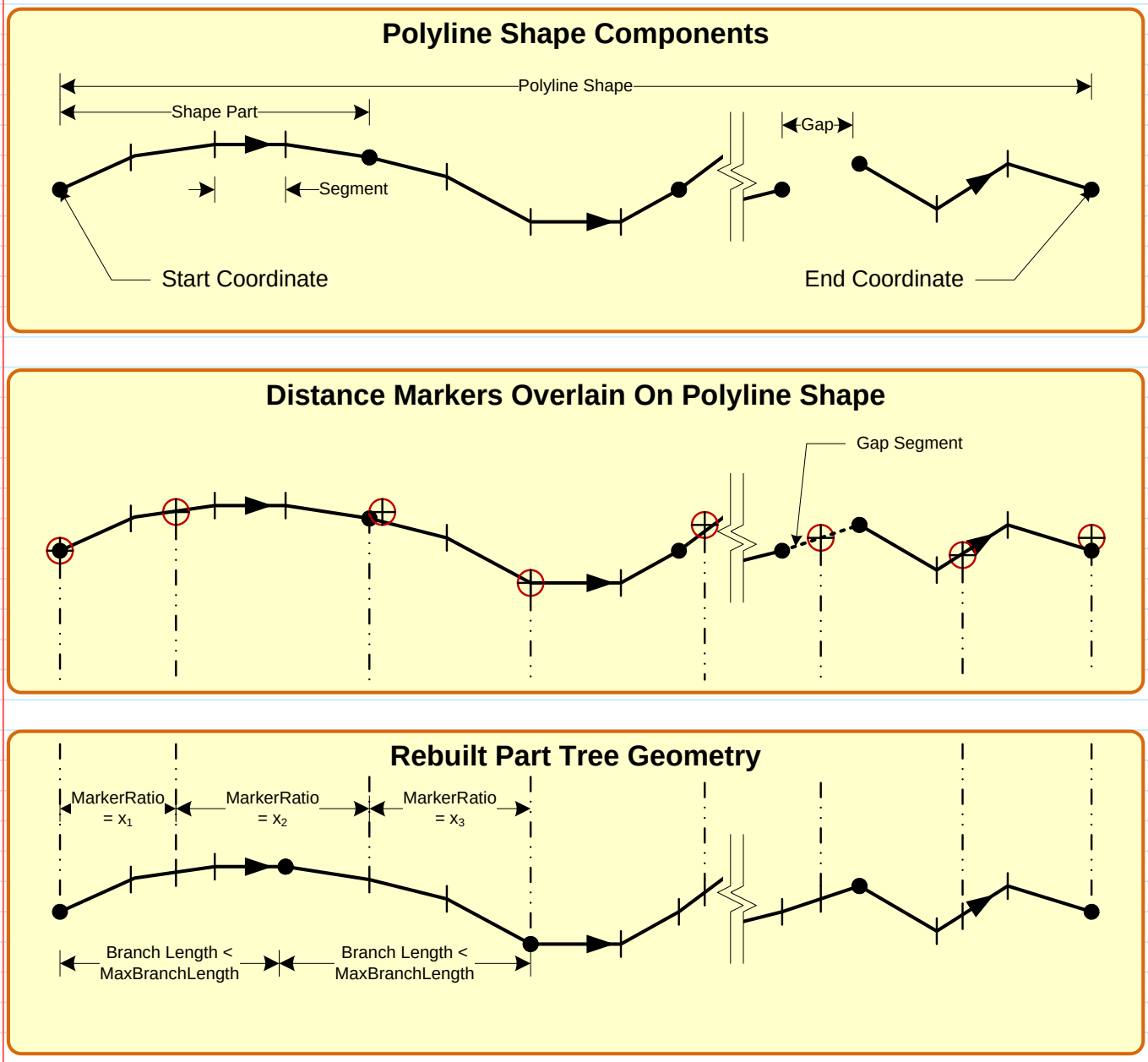


Figure 1. Illustration of the Part Tree Build Process Using Polyline Shapes and Distance Markers

LM Calc Search Optimization

Wednesday, March 20, 2013
4:12 PM

Introduction

The search location records often include data entry errors (e.g., incorrect coordinates or search path identifiers). Without any simple checks, the return Marker Distances can be unrealistically large. For this reason, a set of Search Buffers is used to identify the potentially incorrect Marker Distance matches and avoid obvious data errors. They are also used to optimize the search process. The following is an overview of how Search Buffers are used to optimize the Part Tree Search Process.

Search Buffers

As shown below, a Search Buffer is defined as a maximum distance from the Search Path's polyline shape. The following two Search Buffers are used in the Marker Distance calculation:

- The ***On-Line Search Buffer***: For locating valid ***On-Line*** match points (i.e., match points located between the Part Tree's Root and End Branches).

- The ***End-Node Search Buffer***: For locating valid ***Off-Start*** match points (i.e., match points located within the search buffer distance from the Levee's Root Node, but not On-Line) or ***Off-End*** match point (i.e., match points located within the search buffer distance from one of the Levee's End Nodes, but not On-Line).

If the search point is located outside the Search Buffer, the Match Point's Marker Distance is recorded as an error code. The following four error codes are used:

- ***Missing Segment Error (-999.99)***: Matching Search Path (i.e., Levee or River) could not be located - typically, because the search point is located outside the bounds of the Search Path.

- ***On-Line Error (-999.88)***: An *On-Line* match was found, but the inspection location is outside the *On-Line Search Buffer.*

- ***Off-Start Error (-999.77)***: An *Off-Start* match point was found, but the distance between the Levee's Root Node and inspection location is larger than the *End-Node Search Buffer*.

- ***Off-End Error (-999.66)***: An *Off-End* match point was found, but the distance between the Levee's End Node and inspection location is larger than the *End-Node Search Buffer*.

Using Search Buffers to Optimize Searches

Search for a specific match point along the search path can be optimized using the specified Search Buffers as follows:

1. The fist step is to check if the Specified Location is within the LM Calc's Search Buffer, which is defined as the spatial bounds of the LM Calc's geometry expanded to twice the On-Line Search Buffer size. If the location falls outside these bounds, the reported Match Point Type is *Out-of-Bounds* (e.g., point **MP10** in the figure below).

2. The next step is to select the Search Branch that contains the Specified Location. Similar to the above approach, the Search Branch's Search Buffer is set to the branch's geometry bounds expanded to twice the On-Line Search Buffer.

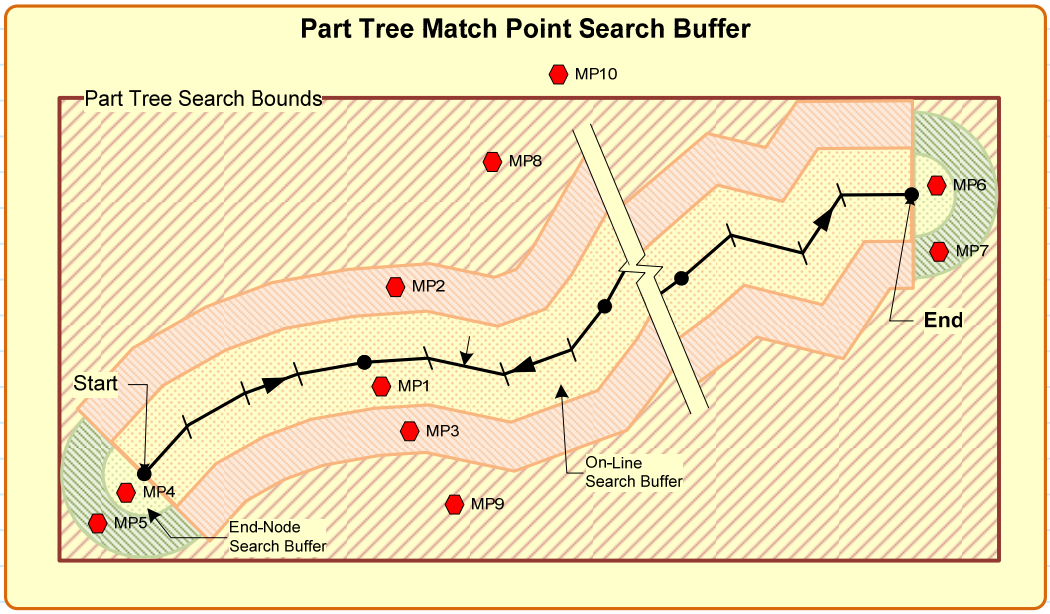
3. For multiple Search Branches that contain the Specified Location within their Search Buffers, the next step is to locate the Search Segment with the closest Match Distance.

LM Calc Search Optimization (continued)

Friday, August 16, 2013
9:43 AM

Using Search Buffer to Optimize Search Optimization (continued)

4. If a Match is found, the *Match Type* will be defined based on whether the Match Distance is smaller than the Search Buffer, less than twice the Search Buffer, or more than twice the Search Buffer. Also, if outside the Search Buffer but within the Part Tree's Search Bounds the *Match Point Type* can be used to determine whether the mismatch location is *OnLine, PastP1*, or *PastP2* (as illustrated in the figure below).



The result matrix below shows typical Match Point results for the points shown in the above figure.

Match Point Result Matrix

Match Point	Match Type	Match Point Type	Marker Distance
MP1	Found	OnLine	Calc'ed Distance
MP2	OffLine	OnLine	-999.88
MP3	OffLine	OnLine	-999.88
MP4	Found	PastP1	Calc'ed Distance
MP5	OffStart	PastP1	-999.77
MP6	Found	PastP2	Calc'ed Distance
MP7	OffEnd	PastP2	-999.66
MP8	NoMatch	OnLine	-999.99
MP9	NoMatch	OnLine	-999.99
MP10	NoMatch	OutOfBounds	-999.99

Search Buffer Size

How big should this Search Buffer be? It is expected that the Specified Location of a known Search Path should be on the Search Path and, most likely, not be more than 100-200ft from the Search Path polyline shape, which is representing the Search Path's center line. If the reported location is outside this buffer it is most likely due to one of the following reasons:

a. The Search Path was entered incorrectly.
b. The specified location's coordinates were entered incorrectly.
c. The specified location is beyond the bounds of the Search Path polyline.

All the above cases should be reported as errors and should be resolved. In all cases, the search buffer will depend on the accuracy of the search path polyline and sensitivity of the search location data.

Finding Match Points of A Specified Location

Wednesday, March 20, 2013
4:23 PM

A Match Point is defined as the point on a segment in the Search Path (i.e., on one of the Search (or Part Tree) Segments of a Search (or Part Tree) Branch) that represents the Specified Location, which is the point along the search path geometry that is closest to the Specified Location. The Marker Distance is calculated from the start of the Search Tree (i.e., from the start of the Tree's Root Branch) to the match point along the Search Path.

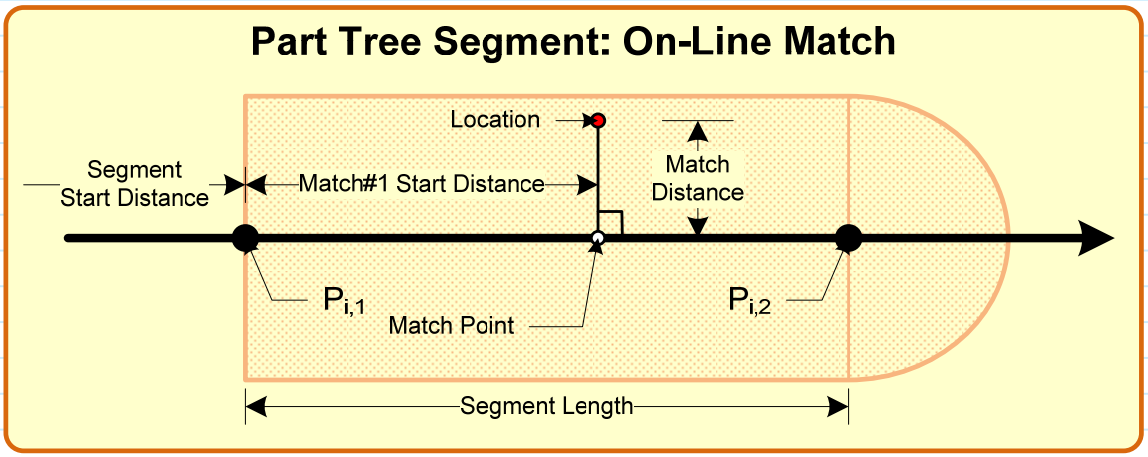
A Match Point has two properties:

- ***Match Distance*** = the shortest distance between the Specified Location and the Search Segment .
- ***Start Distance*** = the Marker Distance along the Search Path from the Root Branch's Start Point plus the Start Point Marker Distance offset.

The methodology used in calculating the Marker Distance for a specified Inspection Location along a defined Search Path finds all Match Points along the path within the specified Search Buffer. The Match Point with the shortest Match Distance is selected to return the Match Point's Start Distance as the Specified Location's Marker Distance.

The following is an overview of the how Match Points are calculated.

Case 1: Identify An On-Line Match Point



If the Specified Location is parallel to a Search Segment, the shortest distance between the Specified Location and the line segment is calculated as the length of the line through the Specified Location perpendicular to the Search Segment and the match point is found as the point where the perpendicular line intersects the Search Segment .

The Match Point's Start Distance is calculated as the Search Segment 's Start Distance plus the segment's *Marker Length* (i.e., the distance between the Segment's start point and the Match Point multiplied by the Search Segment's *MarkerRatio*).

NOTE:

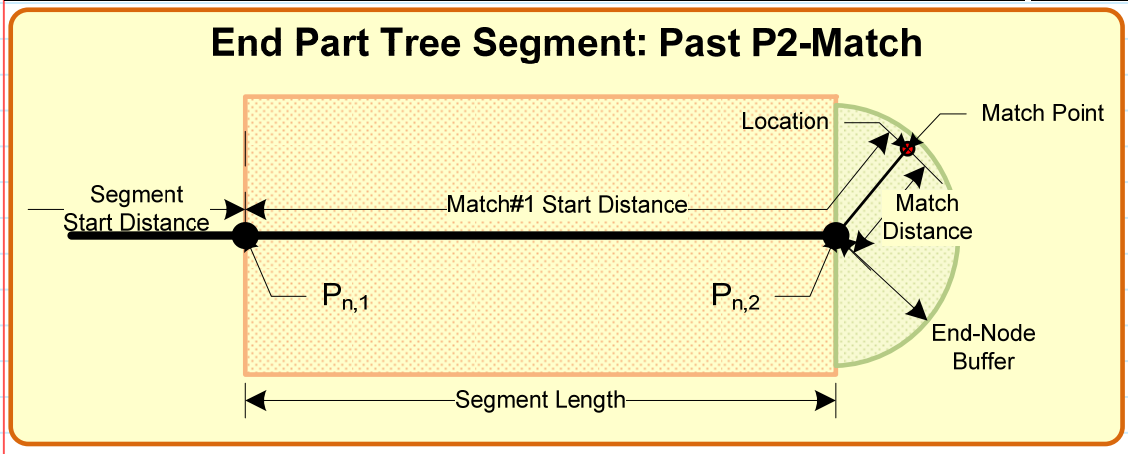
a. A Search Segment's Start Distance = (the Root Branch's Start Offset) + (the sum the Marker Lengths of all parent Search Branches) + (the sum the Marker Lengths of all parent Search Segments), or in other words, the total Marker Distance from the start of the Search Path to the start of the Search Segment plus the Root Branch's (optional) Start Offset.

b. The *Marker Length* of a segment is calculated as the length of the segment multiplied by the segment's *MarkerRatio*. The Marker Length of a Search Branch is the sum of the Marker Lengths of all its segments.

Finding Match Points of A Specified Location (continued)

Wednesday, March 20, 2013
4:48 PM

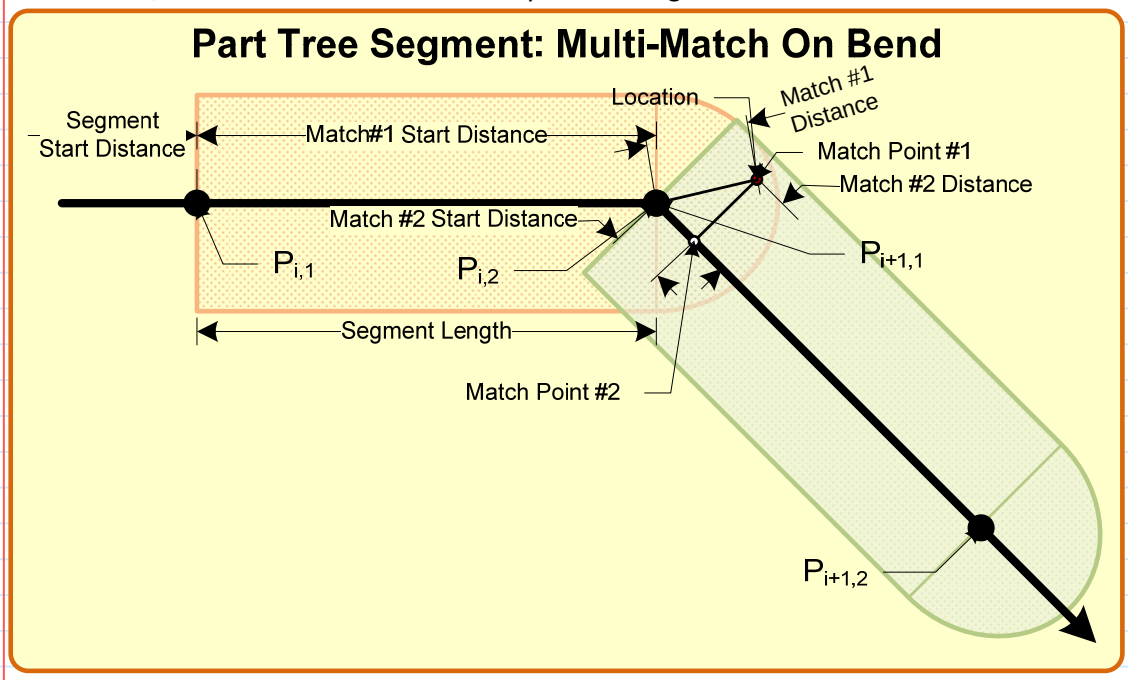
Case 2: Identify a Match Point for the End Segment if the Location is Past the Search Segment 's End Point (P2)



If the Specified Location is past the end point ($P_{n,2}$) of the End Search eBranch's End Segment (i.e. past the End Point of the Search Tree, but still within the End-Node Buffer), the Location is considered a Match Point with the Match Distance as the distance between the Segment's end point and the Specified Location and the Match Point's Start Distance as the Match Distance times the Segment's *MarkerRatio* added to the Segment's End Distance (i.e., the Segment's Start Distance plus the Segment's length).

Case 3: Identify a Match Point for two Adjacent Segments if the Location is Past the First Search Segment 's End Point (P2)

As illustrated in the figure below, two adjacent Search Segments could allow two Match Points: an On-Line Match for one Segment; and a Past-P2 Match for its parent segment. In this scenario the Match Distance for Match Point#1 (i.e. the Past-P2 Match Point for the parent segment) is the distance from the Segment's End Point ($P_{i,2}$) to the Location; but its Start Distance will always be the Segment's End Distance.



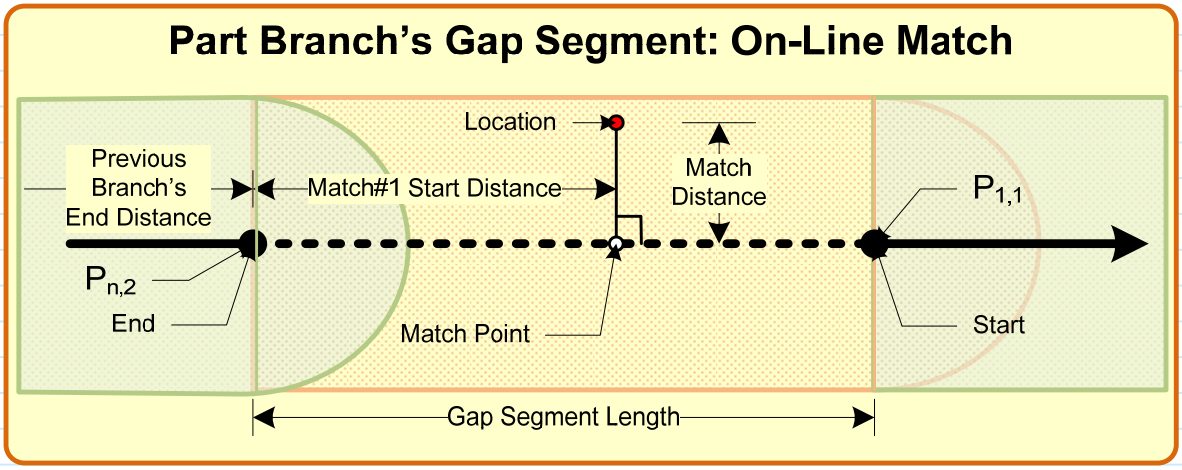
The Match Point for the On-Line Match Point (Match Point#2) will be calculated as described in Case 1, because it has the shorter Match Distance.

NOTE: On a bend (as shown in the figure)there is a window where there is a Match Point#1, but no Match Point#2.

Finding Match Points of A Specified Location (continued)

Wednesday, March 20, 2013
5:28 PM

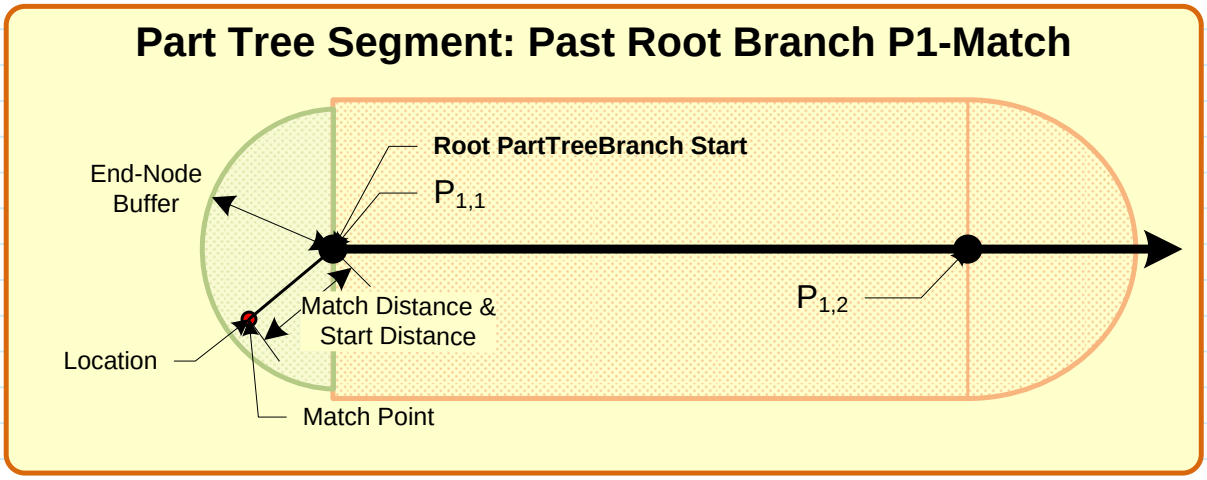
Case 4: Finding Matches in a Gap between Two Search Branches



If there is a gap between two Search Branches, and this gap should be considered in the Marker Distance calculation, a "dummy" *Gap Segment* is added to fill the gap, and the On-Line Match Point is handled as shown in the figure above. The Match Point's Start Distance is the prior Search Branch's End Distance plus the distance from the Node's End Point ($P_{n,2}$) times the Gap Segment's *MarkerRatio* to the Match Point.

If the gap should not be considered and the location is in the gap, the Match Distance is calculated as the distance between the prior Search Branch's end point ($P_{n,2}$) and the Location, and its Start Distance (Levee Mile) is the parent Search Branch's End Distance.

Case 5: Finding Matches Before the Root Node's Start Point (P1).



If the Specified Location is before the start ($P_{1,1}$) of the First Segment of the Root Search Branch (see figure above) and within the End-Point Search Buffer, the match point is the Inspection Location and the Match Distance is the distance between the Segment's Start Point ($P_{1,1}$) and the Match Point and the Start Distance is the Root Search Branch Start Offset minus the Match Distance.

NOTE: In this case the Match Point's Start Distance can be negative.