PART I: 2D IPO's FOR CONSTANT SPEED LINES – CURVES – NURBS

Valere Huypens, Member IEEE

Aen den Hoorn, App. 8 bus 1, 3440 Zoutleeuw, België
email: huypens@telenet.be

Abstract

Current constant speed IPO's, usually, use Sampled-data IPO's and constant speed lines use the wrong initialized software DDA-ipo's, which make these IPO's unusable. The Bresenham- and midpoint IPO's are non-constant speed reference pulse IPO's with bounded inaccuracy. By adding an ultra-fast 3-lines algorithm "PRM-cs" to the actual midpoint or Bresenham algorithms, we convert these midpoint-ipo's to very fast, constant speed, reference pulse IPO's. This applies to 2D-lines, 3D-lines, 2D-curves and 2D-NURBS. The PRM-cs measures, in real-time, the length of the discrete curve and the PRM-cs is completely new.

We define the best IPO, the major axis principle and the LSD-priority.

The major axis principle holds for the actual 3D-line IPO's. These IPO's are, generally, imperfect, but they can be updated to constant speed 3D-line IPO's, when the production manager agrees.

The Digital Geometric Geometry (DAG) defines the discrete lines globally, but this global definition of a discrete 3D-line, gives discrete 3D-lines whose accuracy is much less than the accuracy of the best discrete 3D-lines (e.g. 37% worse).

We describe the three causes of the imperfect discrete 3D-lines.

All existing pulse-rate or PRM-ipo's use a wrong initialization, which deteriorates the accuracy. We determine the right initialization for the new PRM-cs and the updated PRM-ipo.

We propose the benchmark-ipo "listSIM-ipo". This constant speed IPO can, also, be used in real-time for every 2D- and 3D-curve.

The 3rd-degree Trident NURB shows that the constant speed reference pulse method is much better than the existing sampled-data methods.

Keywords

Bresenham's algorithms; Midpoint method; two-point-method; Discrete curves; NURBS Interpolation; Software DDA; pulse-rate multiplier; bit-rate multiplier; Diophantine inequality; Global definition of a discrete line; Best 3D-line; CNC-interpolation; CNC-machining.

Abbreviations

Alaska: programming language Alaska Xbase++ version 1.9 from Alaska Software, Germany;
 BLU: basic length unit, mostly in micron meter;
 BRES-ipo: the collective noun of the MIDP- and 2PRES-IPO;
 BRM: binary rate multiplier;
 BRM-ipo: linear DDA IPO, software DDA IPO;
 Cost: the sum of the squared errors of the discrete points for a curve;
 DAG: digital analytic geometry [12];
 DAG-ipo: non-constant speed major axis IPO based on the DAG;
 DDA: digital differential analyzer [19];
 dErr[P_n]: absolute perpendicular distance of the point P_n to a curve, also called the error;
 EDM: electrical discharge machining;
 Extreme point: tangential point with tangent parallel to a main axis;
 GPU: graphical processor unit;
 HSM: high-speed machining;
 IPO: interpolator;
 LSD: least square distance;
 LSD-ipo: the collective noun of the VIRT- and BRES-ipo;
 Mathematica: the technical computing system of Wolfram Research Illinois, version 10.0.2;
 MaxErr: the maximum error of the discrete points for a curve;
 MinMaxErr: the minimum MaxErr of a discrete line;
 MIDP-ipo: non-constant speed IPO based on the pairwise midpoint comparison method;
 NURB: non-uniform rational B-spline;
 OoC: out-of-control [5, Chapter 6];
 OoA: out-of-accuracy [5, Chapter 7];
 PRM: pulse rate multiplier
 PRMS-cs: converts the LSD-ipo's to constant feedrate IPO's;
 PRM-ipo: non-constant speed major axis IPO without division or multiplication;
 QSIC: quadratic surface intersection curve;
 VIRT-ipo: non-constant speed IPO based on selecting the candidate point with the LSD with integrated priority;
 2PRES-ipo: IPO based on the pairwise two-point residue comparison method;

Mathematical Style

The equations are written and checked with “Mathematica”. To obtain the meaning of an unknown function, google: “Mathematica and <the name of the unknown function”. For the scientific style, we used the Standard Form instead of the Traditional Form of Mathematica, therefore the reader can reproduce the algorithms and check the results.

1. Introduction

1.1 Speed – position closed-loop control systems

This paper describes the software IPO's for discrete lines and curves of devices that must (physical) control precisely the movements of these devices with constant instantaneous velocity along the curve. There exist two methods [9] & [11] to implement the IPO's in a velocity-position closed-loop system: the Reference Pulse Method (Fig. 1) and the Sampled-Data Method (Fig. 2).

Fig. 1 Velocity-position system based on the reference pulse IPO

Fig. 2 Velocity-position system based on the sampled-data IPO

It is also possible to use no feedback, but in closed-loop systems, the up-down counter contains the servo error, which increases/decreases when the velocity increases/decreases, therefore the acceleration/deacceleration is soft when the servo is fine-tuned. The maximum servo error is double bounded, the first bound stops the input and the second bound gives an alarm. Therefore, the closed-loop is safe and the acceleration/deceleration is bounded.

We will not use open-loop systems, although they still exist for simple systems. A well-known example is the computer control of a plotter actuated with stepping motors. In that case, one does not want a constant velocity along the curve, but one must use an

acceleration/ deacceleration profile or limit the frequency F_c , such that no pulses are lost. The IPO used, was Bresenham's (two-point method) algorithm for lines [1] and circles [2]. Bresenham's algorithms are non-constant speed, very fast BRES-ipo's (§1.5), without integrated priority, and they can also be used in closed-loop systems based on the reference pulse IPO. Using the Pulse-Rate-Multiplier PRM-cs (§1.6 & §1.8) we will convert these algorithms into very fast constant-speed-algorithms.

1.2 The cubic and square grid describes the output of the axes of velocity-position systems

To describe the output of 3D-curves, we use a cubic grid as defined in Fig. 3 and to output 2D-curves we use a square grid defined in Fig. 4. The grid distance along the axes equals one BLU (resolution) or Δ (in the equations), although, in practice, Δ equals one, we will further use the symbol Δ in the equations because it makes the equations clearer.

A cubic grid defines the grid points in a right-handed three-dimensional Cartesian coordinate system. A discrete curve travels along the grid points, and we call the process of finding the discrete points the IPO. The curves using the reference pulse method must be k-connected. When we don't mention the connectedness of a curve, then the 2D curve is 8-connected and the 3D-curve is 26-connected, hence the default k-value is 8 or 26 (in the Digital Analytical Geometry these curves are 0-connected [12]).

$$P_B=P_1, P_C=P_3, P_D=P_2, P_M=\frac{P_1+P_3}{2}, P_H=\frac{P_2+P_1}{2}, P_V=\frac{P_2+P_3}{2}, P_h=\frac{P_A+P_3}{2}, P_v=\frac{P_A+P_1}{2}$$

Fig. 3 "3D"

Fig. 4 "2D"

1.2.1 Constant feedrate – Constant speed

In physical systems (machine tool practice, EDM, 3D-printing, ...), the word "speed" refers to two separate velocities, for example for milling, the cutting speed or surface speed is the relative velocity between the cutting tool and the surface of the workpiece (m/min). The feedrate is the relative velocity at which the cutter is advanced along the workpiece (mm/min). In that case, constant speed means the continuous constant feedrate, in the center of the tool, along the curve. In general, constant-speed IPO's don't include the automatic feedrate adaption as used in "Circle Milling": when the radius of the surface equals R_s and the radius of the cutter equals R_c , and the finishing feedrate along an axis equals V , then the feedrate for outer finishing must be reduced by $V * \frac{R_s - R_c}{R_s}$ and the

feedrate for inner finishing must be increased by $V * \frac{R_s + R_c}{R_s}$. Our method can take care of

this speed adaption when R_c and the inner or outer finishing is given; the radius of curvature R_s must be calculated by the IPO. For inner finishing, the PRM-ratio LM/LN (§1.8) becomes

$\frac{LM}{LN} * \frac{R_s + R_c}{R_s}$, but this ratio must be smaller than one; in that case, the input frequency must

be augmented. Although the “Die & Mould Making Application Guide of Sandvik Coromant gives maximum feedrate values of 40 – 60 m/min for HSM (High-Speed Machining), our NURB references [13] & [14], gives maximum feedrates of 1.5 – 12 m/min for resolutions of 0.5 – 7 μm .

Notice that the dynamics of the drive system does not ensure the constant speed along the curve, but that the IPO must provide the constant speed. So, figures 1.1 also represent non-constant-speed velocity-position systems, and we have constant-speed-IPO's and not-constant-speed IPO's

1.2.2 Reference-pulse-Systems

As you will see in the sequel, we have to review all the reference pulse IPO's with the maximum accuracy less or equal than the upper bound (Table 1). We hope that the classification is helpful.

Fig. 5 The classification of the non-constant speed reference IPO's

In reference-pulse-systems, we subdivide the curve into monotonic segments, characterized by its *monotonic direction vector* $\{S_x, S_y, S_z\}$ and its cubic set $\text{set}_{3D} \triangleq \{\delta_x, \delta_y, \delta_z\}$; each S_x , S_y and S_z equals ± 1 and each δ_x , δ_y and δ_z equals Δ or 0. The cubic $\{\Delta_x, \Delta_y, \Delta_z\}$ equals $\{S_x * \delta_x, S_y * \delta_y, S_z * \delta_z\}$ and in practice Δ equals one. In this way, the direction vector for a monotonic segment is fixed, but the cubic can collapse.

In systems based on the reference IPO, the actual best gridpoint P_n is the basepoint P_A of the 2D candidate grid points P1, P2, P3 or the 3D candidate grid points P1, P2, P3, P4, P5, P6, P7 of the next best point. The best gridpoint is the candidate point with the minimum perpendicular distance to the curve; hence *the discrete curve travels along the grid points with minimum perpendicular distance to the curve*.

The reference IPO can generate single pulses x, y, or z, double pulses xy, yz, or zx and a triple pulse xyz. The interrupt frequency $F_c = \frac{1}{T_c}$ determines the speed of the axes and the generated pulses are directly forwarded to the corresponding up-down counters. One x (y, z) pulse denotes one BLU (Basic Length Unit) of the x-axis and corresponds with the grid distance Δ_x . The reference IPO uses internally the grid distance $\Delta = \Delta_x = \Delta_y = \Delta_z = 1$, but the real grid distance corresponds with BLU μm . The accuracy of reference-pulse-systems is a fraction of Δ or one BLU, but the maximum execution time T_c of the reference pulse-ipo limits the velocity.

Sampled-Data-Systems:

The sampling time T_s can be some 100-times greater than reference interrupt time T_c . At the input, the sampled-data system can be described in the following way. During the period T_c the reference-pulse-system generates the incremental moves pulse[nkey] and during the period T_s , the sampling program can put the T_c pulses in the sampling word, hence the sampling word[i]=sampling word[i-1]+pulse[i-1], or the sampling word[T_s] equals the integration of the pulses over T_s . So, sampling-data systems use Euler- or Taylor- or Tustin-integrators [11]. When you look to the dynamics (output) you can use the rules of z-sampled data systems or you can consider the sampled system as a continuous system with delay $\frac{T_s}{2}$ and apply the integration. It is well known that sampling makes the dynamics of the system less stable, therefore the amplification is lower, and the accuracy is lower. At the input, a sampling-data system never works in this way: the T_c pulses between the sampling period T_s are not measured, so one only measures the values at the sampling instants T_s , and the regulated sampled-data system will have very well results at the sampling instants, but the dynamic behavior can be quite different between the sampling instants. So, the accuracy of sampled-data systems is high at the sampling instants, but in-between the accuracy must be bounded, or one must use subsampling ("fine line interpolation") [11, pp 96 3.4]. In conclusion, we can say that the sampled-data systems are faster, have less accuracy and are more complex. Therefore, we try to avoid sampled-data systems, although we must compare our NURBS-reference-pulse-algorithms with existing NURBS-sampled-data-algorithms (§2).

The implementation of Reference-Pulse-Systems is simpler than that of Sampled-Data-Systems; hence we need to find very fast constant-speed-reference-pulse-ipo's.

1.3. The best IPO and The Least-Squared-Distance IPO's or LSD-ipo's

The absolute perpendicular distance of a point P_n to a curve is the error $dErr[P_n]$. The perpendicular distance $dErr[P_n]$ is different for the sets: lines, circles, conics, NURBS. The LSD criterium is the same as the minimum absolute perpendicular distance criterium. The LSD-ipo creates the k-connected grid points $\{P_n\}$ and sometimes, the corresponding array $listErr \equiv \{ Abs[dErr[P_n]] \}$, $n \in \{0, \dots, k_{MAX} - 1\}$.

This array defines the important parameters of the k-connected curve

- $\text{MaxErr} \equiv \text{Max}[\text{listErr}] \equiv \text{Max}_{k=1}^{k_{\text{MAX}}} \{d\text{Err}[P_n]\} ;$
- $\text{Cost} \equiv \text{Dot}[\text{listErr}, \text{listErr}] \equiv \sum_{k=1}^{k_{\text{MAX}}} d\text{Err}^2[P_n] .$

When, for some IPO's and for all objects of its set, the $\text{MaxErr}[\text{object}]$ equals the minimum $\text{MaxErr}[\text{object}]$ and for one of the IPO's, the $\text{Cost}[\text{object}]$ also equals the minimum $\text{Cost}[\text{object}]$ then that IPO is perfect, and we call it the best IPO.

An inaccurate IPO is imperfect when for some object the MaxErr is not minimum, but the

MaxErr is bounded (Table 1: the upper bound is $\frac{\Delta}{2}$ for 2D, $\frac{\sqrt{2}}{2}\Delta$ for 3D) .

In that sense, we can loosely define the best IPO as:

The best IPO is the IPO with the minimum MaxErr and for two or more IPO's, with the same minimum MaxErr, the IPO with the minimum Cost is the best.

We must use monotonic segments when we use the LSD-ipo, because,

- We can detect OoC (Out-of-Control) [5] easily. As we know from [5] the point P_n can be OoC when the direction of the tangent in the footpoint of P_n on the curve is not in accordance with the monotonic direction vector, and in that case, the LSD measurement is not valid.
- For a non-monotonic segment, we must detect the extreme points online, and that is quite complex.
- It limits the number of candidate points to 3 in 2D (Fig. 4) and 7 in 3D (Fig. 3).
The midpoints of these points are very important, i.e. the 2D-midpoints P_M, P_H, P_V .

1.4. Definition of the major axis of a line, the priority and the LSD-priority

The major axis of a 2D line $\{L_i, L_j\}$ or a 3D line $\{L_i, L_j, L_k\}$ is the axis which corresponds to the maximum absolute component of the line. For example, the line $\{4,3\}$ is a major axis x-line and the line $\{49,74,82\}$ is a major axis z-line. For curves, the components of the tangent line determine the major axis, and for quadratic curves, the components of the polar in the midpoint can be used, provided that the midpoint is very near to the curves.

Definition of the major axis principle:

“when the major axis principle holds, the LSD-ipo makes a major axis move at each move.”

We define the priority,

- for the cubic with basepoint P_A , the points P_1, P_3, P_5 have priority one, the points P_2, P_4, P_6 have priority two and the priority of P_7 is three;
- for the square, P_1 and P_3 have priority one and P_2 has priority two.

Let's agree about the next meanings:

- “the major axis line” does not mean that the major axis principle, a priori, holds. For a 2D-line it holds, but for a 3D-line, it generally, does not hold. So, the major z-axis 2D-line always makes a z-move, but that is not the case for a major z-axis 3D-line.
- “the major axis IPO” means that the generation of the discrete line applied the major axis principle. So, all actual 3D-IPO's are major axis IPO's and all 2D-IPO's are major axis IPO's (§1.8), but there also exists a non-major axis 2D-IPO, e.g. the LSD- and BRES-ipo; in

that case, the major axis principle holds, but the algorithm does not use that fact. So, most published BRES-ipo's are major axis 2D-IPO's and the IPO of [6] is a non-major axis 2D-IPO. The actual published 3D-line IPO's are all major axis 3D-IPO's (§1.7.3).

Definition of the LSD-priority:

We apply the LSD-priority when we select the candidate gridpoint of a k-connected curve with the LSD with integrated priority:

The LSD-point is the candidate gridpoint with the minimum absolute perpendicular distance to the curve. The LSD-point is chosen, but when two or more points have the same LSD-value then the point with the highest priority is chosen, but when the LSD-points have the same priority and when the major axis principle holds, we could select the point along the major axis direction. But for most 3D-lines, the major axis principle does not hold (Table 3) and the calculation of the right major axis for curves is complex, as it is the tangential direction in the footpoint (for quadratic curves one could approximate this tangent as the tangential direction of the polar of the midpoint). When we use the BRES method, we do not use a major axis, because this method uses the result of “all” the midpoints $\{P_M, P_H, P_V\}$ [6]. Applying the “LSD-priority” makes the k-connected curves unique.

1.5. Definition of the LSD-ipo's: the VIRT-ipo, the BRES-ipo {MIDP-ipo, 2PREP-ipo}

We can use two LSD criteria, using the integrated priority, for monotonic segments:

- the dLSD (direct Least Square Distance) method: we calculate the Square Distance of each candidate point and select the candidate point with the LSD with integrated priority. This method is very slow for non-linear curves and its non-constant-speed algorithm is called the VIRT-ipo (Virtual-ipo) because it cannot be applied in real-time;
- the pairwise MIDP comparison method: we compare the Square Distances of the candidate points, pairwise to the polar line of the midpoint of these candidate points until the best candidate point is found. The polar line of the conic $\text{Dot}[\mathbf{P}, \mathbf{G}] + W \equiv 0$ [4, 5] of the midpoint $\mathbf{P}_M = \{x_M, y_M\}$ is $X_M * x + Y_M * y + W_M \equiv 0$. The signed distance of a point

$\mathbf{P}_n = \{x_n, y_n\}$ to the 2D-line is $\rho_n = \text{sErr}[\mathbf{P}_n] = \frac{X_M * x_n + Y_M * y_n + W_M}{\sqrt{X_M^2 + Y_M^2}}$. When ρ_i^2 (i=C or B Fig.

4) is the square distance of the point $\mathbf{P}_i$, then the comparison $\rho_C^2 > \rho_B^2$ can be written as $d_M = -2 * S_{Lxy} * F_M[x_M, y_M] > 0$ [5, pp 24, 25], $S_{Lxy} \equiv S_{\text{LEFT}} * S_x * S_y$, $F_M[x_M, y_M]$ is the residue of the conic or the line in the midpoint $\mathbf{P}_M = \frac{\mathbf{P}_C + \mathbf{P}_B}{2}$ and S_{LEFT} equals 1 / -1 when the residue is negative/positive on the left side of the line or conic, hence S_{Lxy} is a fixed constant in the monotonic segment $\{S_x, S_y\}$.

Analogous, $\rho_D^2 > \rho_B^2$ corresponds with $d_H = -2 * S_{Lxy} * F_H[x_H, y_H] > 0$, and $\rho_C^2 > \rho_D^2$ corresponds with $d_V = -2 * S_{Lxy} * F_V[x_V, y_V] > 0$ [5, pp 31, 32].

For two points with the same LSD, $\rho_C^2 \equiv \rho_B^2 \Leftrightarrow F_M \equiv 0$, we could select P_B when the x-axis is the major axis, else we could select P_C . But as shown in [5], the midpoint measurement is OoA when the midpoint is on or very near inside the conic, therefore as done in [6], we avoid OoA by regarding the points on the conic as points outside the conic. When $\rho_D^2 \equiv \rho_B^2 \Leftrightarrow F_H \equiv 0$ or $\rho_C^2 \equiv \rho_D^2 \Leftrightarrow F_V \equiv 0$, we use the integrated priority,

hence we select P_D . Instead of using the major axis, the candidate points P_B , P_C or P_D can better be selected from $\text{Sign}[\{-S_{Lxy} * F_M, -S_{Lxy} * F_H, -S_{Lxy} * F_V\}]$ [6, pp 6-8],[5,Ch3] when the measurement is not OoC.

This method can be realized with Boolean variables; hence it is very fast. The algorithm based on this criterium is called the MIDP-ipo. For non-quadratic implicit curves, we can instead of F_M use the sign of the arithmetic mean decision variable F_{aM} of two candidate points [5, pp 21]. The algorithm based on this criterium is called the (Two Point Residue) 2PRES-ipo; this IPO can also be applied to conics but in that case, we proved [5] that it can have more OoA points [5]. The MIDP- and 2PRES-ipo's are identical for lines and faster for quadratic curves because the decision variables are based on the distance of the midpoints to the polar lines of these midpoints. The midpoints are the points P_M , P_H and P_V for conics, but we have 21 midpoints for QSIC's (Quadratic Surface Intersection Curve). A very fast non-constant-speed IPO is given in [6] and in Part III we will give constant-speed IPO's for QSIC's.

This method is very fast and its non-constant-speed real-time algorithm with integrated priority is called the BRES-ipo.

The VIRT-ipo, with integrated priority, is the collective noun of the non-constant-speed reference IPO using the direct Least Square Distance method.

The BRES-ipo, with integrated priority, is the collective noun for the MIDP- or the 2PRES-ipo. Generally, these IPO's use the LSD with integrated priority, but most of the time, the published IPO's are implemented without the integrated principle and often one presupposes that the major axis principle holds. By default, we will use the integrated priority and then, we don't care about the major axis principle.

An IPO which does not use the major axis principle, a priori, must not subdivide the monotonic segment into major axis subsegments, and that is a great advantage.

The LSD-ipo's is the collective noun of the VIRT- and BRES-ipo's and these IPO's can have OoC-measurements, and in that case, the measurement is not valid, but this situation can be detected easily, and the solution is very simple [5],[6]. Opposite to the VIRT-ipo's, the BRES-ipo's can be OoA when the midpoint is inside and very near to the curve [5, Appendix 4: point {4,2} of the circle].

Summarized:

The VIRT-ipo has no OoA, can be applied to implicit and parametric curves, but, generally, the time needed to calculate $dErr[P_n]$ is often complex and long, hence the VIRT-ipo is generally slow, but the BRES-ipo's are very fast.

The MIDP- and 2PRES-ipo can have OoA-points, the MIDP-ipo can be applied to quadratic curves and lines, and the 2PRES-ipo can be applied to implicit curves.

The LSD-ipo's can have OoC-points and these points cannot be used. It is not necessary that the major axis principle holds for the LSD-ipo's.

The major axis principle holds for most published 2D-lines and all the published 3D-lines, and this simplifies incremental 2D- and 3D-algorithms. In paragraph 1.8 we shortly treat these 2D and 3D IPO's: DAG-, Bresenham- and PRM-ipo.

1.6. The ultra-fast **PRM-cs** algorithm converts the **LSD-ipo** into a constant-speed **IPO**

The PRM-cs is valid for 2D-lines $L=\{L_i, L_j\}$ and the actual major axis 3D-lines $L=\{L_i, L_j, L_k\}$.

L is the direction vector and we initialize (§1.8) the algorithm with $LM_{cs}=\text{Max}[\text{Abs}[L]]$,

$LN_{cs}=\text{Norm}[L]$, $AC_{cs} = \frac{LN_{cs}}{2}$, $npuls=nkey=0$, and we start the interrupt with frequency $F_{in} = \frac{1}{T_{in}}$.

```
While[ endpoint is not reached,
  Label[StartCS];
  Wait for interrupt  $T_{in}$ ;
  npuls++;  $AC_{cs}=AC_{cs}-LM_{cs}$  ;
  If[ $AC_{cs} > 0$ , Goto[StartCS] ] ;
   $AC_{cs}=AC_{cs} + LN_{cs}$  ;
  nkey++;
  LSD-ipo;
  For curves, eventually update  $LM_{cs}$  and  $LN_{cs}$ ;
]
```

Conditions:

- $\frac{LM_{cs}}{LN_{cs}} \leq 1$, $LM_{cs}>0$, $LN_{cs}>0$, $AC_{cs}>0$ (analogous to the PRM-ipo of §1.8) .
- The execution time of PRM-cs plus LSD-ipo $< T_{in}$ and hence the execution time of the LSD-ipo determines if the LSD-ipo can be applied in a reference pulse constant speed system.
Therefore, we must replace the LSD-ipo with the BRES-ipo.
- The speed ratio of the PRM-cs is the reverse of the tangential speed ratio of the LSD-ipo; hence the LSD-ipo must, for curves, communicate the direction vector to the PRM-cs.

Notes:

- $\text{Norm}[\{L_i, L_j, L_k\}]$ equals $\sqrt{L_i^2 + L_j^2 + L_k^2}$,
- The PRM-cs does not change the discrete curve, so it has no influence on accuracy. Hence the MaxErr and the Cost are independent of the PRM-cs,
- The LSD-ipo calculates the grid points $\{x_n, y_n, z_n\}$,
- For curves, $\{L_i, L_j, L_k\}$ is a function of the grid points and its footpoints,
- $\{LM_{cs}, LN_{cs}\}$ depends on the connectedness of the line or curve (Table 1),
- “ $AC_{cs} > 0$ ” corresponds with the Ceiling-form, but “ $AC_{cs} \geq 0$ ” can also be used,
- $npuls$ is very sensitive to the initialization of AC_{cs} , therefore, the fast simplified square root algorithm [IEEE computer] can only be applied for an approximately constant speed algorithm,
- When we don’t simulate this IPO, we delete $npuls$ and $nkey$,
- When we simulate the IPO, then $npuls$ measures, online, the length of the curve,
- Table 1 makes it clear that the 4- and 6-connected IPO’s make no sense: their accuracy is worse and when using constant speed, the maximum speed is half the maximum speed of the 8- or 26-connected IPO’s. So, the 4- and 6-connected IPO’s are obsolete, although a half-century ago our Ducomat CNC [3] made all the geometrical calculations (intersections, equidistant contours, cut distribution) with 4-connected line- and circle-

ipo's. This system used a PDP-8E computer without floating point hardware. We knew the 8-connected IPO's, but these IPO's cannot be used to calculate the intersections between lines-lines, lines-circles, and circles-circles.

Connected- Ness	2D-line $\{L_i, L_j\}$ 3D-line $\{L_i, L_j, L_k\}$	MaxErr $\leq \text{bound}$	Interrupt Freq F_{in}	LM_{cs}	LN_{cs}
4	2D	$\frac{\sqrt{2}}{2}$	$2 \cdot F_c$	$ L_i + L_j $	$2\sqrt{L_i^2 + L_j^2}$
8	2D [5]	$\frac{1}{2}$	F_c	$\text{Max}[\text{Abs}[\{L_i, L_j\}]]$	$\sqrt{L_i^2 + L_j^2}$
6	3D	$\frac{\sqrt{3}}{2}$	$2 \cdot F_c$	$ L_i + L_j + L_k $	$2\sqrt{L_i^2 + L_j^2 + L_k^2}$
26	3D	$\frac{\sqrt{2}}{2}$	F_c	$\text{Max}[\text{Abs}[\{L_i, L_j, L_k\}]]$	$\sqrt{L_i^2 + L_j^2 + L_k^2}$

Table 1

1.6.1 Main property of simulated npuls = Round[Length of the Curve]

We can prove the next equivalent relations:

$$\text{npuls} = \text{Ceiling}\left[\frac{LN_{cs}}{LM_{cs}} * \text{nkey} - \frac{1}{2}\right] \Leftrightarrow \text{nkey} = \text{Floor}\left[\frac{LM_{cs}}{LN_{cs}} * \left(\text{npuls} + \frac{1}{2}\right)\right]$$

or

$$\text{npuls} = \text{Ceiling}\left[\frac{LN_{cs}}{LM_{cs}} * \left(\text{nkey} - \frac{1}{2}\right)\right] \Leftrightarrow \text{nkey} = \text{Floor}\left[\frac{LM_{cs}}{LN_{cs}} * \text{npuls} + \frac{1}{2}\right].$$

Generally, LM_{cs} is an integer and Δ equals one, and for 2D-lines the LSD-ipo makes a major axis move at each step, therefore the algorithm ends with $\text{nkey} = \frac{LM_{cs}}{\Delta}$, and the first

equivalence relation gives then $\text{npuls} = \text{Ceiling}\left[LN_{cs} - \frac{1}{2}\right] \equiv \text{RoundUp}[LN_{cs}]$.

1.6.2 $V_{\text{tangential}} \equiv V_{in}$ is the main property of the combination of a PRM-cs and an LSD-ipo

For a 2D-line, the PRM-cs gives $\text{nkey} = \frac{LM_{cs}}{LN_{cs}} \text{npuls}$ and $V_{\text{key}} = \frac{LM_{cs}}{LN_{cs}} V_{in}$; the LSD-ipo gives

$$V_{\text{Maj_axis}} = \frac{LM_{cs}}{LM_{cs}} V_{\text{key}} \text{ and } V_{\text{non_Maj_axis}} = \frac{\sqrt{LN_{cs}^2 - LM_{cs}^2}}{LM_{cs}} V_{\text{key}}. \text{ Hence}$$

$$V_{\text{tangential}}^2 = \left(\frac{LN_{cs}}{LM_{cs}}\right)^2 * V_{\text{key}}^2 = \left(\frac{LN_{cs}}{LN_{cs}}\right)^2 * V_{in}^2 = V_{in}^2.$$

1.6.3 The old midpoint- and BRES-ipo's are currently the fastest constant-speed IPO's

Adding the PRM-cs converts the very fast midpoint algorithms into the fastest and simplest reference pulse constant-speed algorithms.

Display, CAD and CAM systems, used or still use the reference pulse not-constant-speed-IPO for lines, circles and conics: [5], [6] extended the midpoint algorithm [4], [7] to the very fast, robust and stable generation of conics. The non-constant speed IPO's, especially the two-point, the midpoint, the scan-conversion of the general conic algorithm (26 pages) were

extensively described in the 2nd edition of Computer Graphics [8]. The criteria used to evaluate these algorithms for a non-monotonic segment of a conic were

1. The midpoint method also called the grid-distance method;
2. The two-point method, also called residual or the arithmetic mean method;
3. The perpendicular distance method to the non-monotonic conic.

As cited in §1.3, we apply the perpendicular distance method to monotonic segments. The midpoint method was chosen as the best. The two other criteria were discarded because they could select a wrong gridpoint, but we proved [5] that all three methods can be OoC and that one must apply the criteria to monotonic segments to detect OoC. The general conics algorithm of [8, 2nd Ed., pp. 951-962] is not stable, we call it berserk because sometimes, the algorithm breaks down and marches away from the conic. This is not a form of aliasing, but the real reason is that the wrong point is OoC, hence the decision variable is not valid, and therefore the algorithm takes a wrong decision. In the 3rd Edition, the midpoint and scan-conversion algorithms do not get any attention, and they justify it in “Differences from the Previous Edition”: “With the advent of the GPU, triangles are converted to pixels (or samples) by radically different approaches than the old scan-conversion algorithms. We no longer discuss those.” So, nowadays, instead of LSD-ipo’s, one uses a GPU dedicated to graphics-related computation. Henceforth, we suppose that you will not find the word “Interpolator” or “IPO”, or constant-speed algorithms in current Computer Graphics and Computer Vision books. The normal conclusion is that the midpoint or two-point algorithms are obsolete, but the opposite is true: we showed that an, at most 3-lines algorithm, called the PRM-cs, converts the “old” very fast non-constant-speed reference pulse IPO’s into “up-to-date” very fast constant-speed reference pulse IPO’s. Generally, constant-speed IPO’s must know the length of the curve, but our method does not need to know the length of the curve, because the PRM-cs measures the length in real-time.

1.7. The current 3D-line IPO’s are imperfect

To keep the description simple, we suppose that the z-axis is the major axis or

$$\text{Max}[\text{Abs}[\{L_i, L_j, L_k\}]] \equiv |L_k|.$$

For all current 26-connected 3D-lines $\{L_i, L_j, L_k\}$, the major axis principle holds.

The major axis principle holds, a priori, for the discrete objects, defined by the Digital Analytical Geometry, hence it’s a fundamental building block of the Digital Analytical Geometry. The idea that the major axis principle must hold for a 3D-line is deep-rooted.

1.7.1 Short Introduction to 3D-lines and imperfect discrete 3D-lines

The 3 projections of a 3D-line can be obtained from $\frac{x-x_s}{L_i} = \frac{y-y_s}{L_j} = \frac{z-z_s}{L_k}$ or from

$$F_x[y_n - y_s, z_n - z_s] = \text{Det} \begin{bmatrix} 1 & 0 & 0 \\ L_i & L_j & L_k \\ x_n - x_s & y_n - y_s & z_n - z_s \end{bmatrix},$$

$$F_y[x_n - x_s, z_n - z_s] = \text{Det} \begin{bmatrix} 0 & 1 & 0 \\ L_i & L_j & L_k \\ x_n - x_s & y_n - y_s & z_n - z_s \end{bmatrix},$$

$$F_z[x_n - x_s, y_n - y_s] = \text{Det} \begin{bmatrix} 0 & 0 & 1 \\ L_i & L_j & L_k \\ x_n - x_s & y_n - y_s & z_n - z_s \end{bmatrix} \text{ or from}$$

$$\begin{pmatrix} F_x[y_n - y_s, z_n - z_s] \\ F_y[x_n - x_s, z_n - z_s] \\ F_z[x_n - x_s, y_n - y_s] \end{pmatrix} = m\Pi \cdot \begin{pmatrix} x_n - x_s \\ y_n - y_s \\ z_n - z_s \end{pmatrix} \text{ with } m\Pi \equiv \begin{pmatrix} 0 & -L_k & L_j \\ L_k & 0 & -L_i \\ -L_j & L_i & 0 \end{pmatrix} \text{ and } \text{Det}[m\Pi] \equiv 0.$$

We also have $F_n = \{F_x[y_n - y_s, z_n - z_s], F_y[x_n - x_s, z_n - z_s], F_z[x_n - x_s, y_n - y_s]\} = \text{Cross}[L, P_n - P_s]$.

Shortly we describe F_n as $F_n \equiv \{F_{nx}, F_{ny}, F_{nz}\}$.

When the point $P_n \equiv P$ belongs to the 3D-line then $\text{Cross}[L, P - P_s] \equiv 0$ as $P - P_s$ is parallel to the line. With the perpendicular vector distance $\rho_n = P_n - P_\perp$, we can write $P_n - P_s$ as

$P_n - P_s = \rho + P_\perp - P_s$ and as $P_\perp$ belongs to the line, we also have $F_n = \text{Cross}[L, \rho_n]$ and

$$\rho_n = \frac{F_n \times L}{L^2} = \frac{\text{Cross}[F_n, L]}{\text{Dot}[L, L]} \text{ after applying the identity } (A \times B) \times C \equiv (A \cdot C) * B - (B \cdot C) * A.$$

$$|\rho_n| = \frac{|F_n|}{|L|} = \text{Norm}[\rho_n] = \frac{\text{Norm}[F_n]}{\text{Norm}[L]} \text{ after applying } (A \times B)^2 \equiv A^2 * B^2 - (A \cdot B)^2, A^2 \equiv A \cdot A$$

The Bresenham distances are $r_{nx} = \frac{F_{nx}}{\sqrt{L^2 - L_i^2}}$ in the $\{y, z\}$ plane, $r_{ny} = \frac{F_{ny}}{\sqrt{L^2 - L_j^2}}$ in the $\{x, z\}$ plane,

and $r_{nz} = \frac{F_{nz}}{\sqrt{L^2 - L_k^2}}$ in the $\{x, y\}$ plane (the latter is not used for a major z-axis 3D-line).

Resuming:

$$F_n = \{F_{nx}, F_{ny}, F_{nz}\} = L \times \rho_n = L \times (P_n - P_s),$$

$$\rho_n = \{\rho_{nx}, \rho_{ny}, \rho_{nz}\} = \frac{F_n \times L}{L^2},$$

$$L \cdot F_n \equiv 0,$$

$$L \cdot \rho_n \equiv 0,$$

$$r_n = \{r_{nx}, r_{ny}, r_{nz}\} = \left\{ \frac{F_{nx}}{\sqrt{L^2 - L_i^2}}, \frac{F_{ny}}{\sqrt{L^2 - L_j^2}}, \frac{F_{nz}}{\sqrt{L^2 - L_k^2}} \right\}.$$

Therefore, we have the following identities

$$L^2 * \rho_{nx} \equiv L_k * F_{ny} - L_j * F_{nz} = \text{Function of } \{r_{ny}, r_{nz}\},$$

$$L^2 * \rho_{ny} \equiv L_i * F_{nz} - L_k * F_{nx} = \text{Function of } \{r_{nx}, r_{nz}\},$$

$$L^2 * \rho_{nz} \equiv L_j * F_{nx} - L_i * F_{ny} = \text{Function of } \{r_{nx}, r_{ny}\}.$$

So, for a major z-axis 3D-line, only the z-component of the 3D-vector distance ρ_n is completely defined using a double $\{r_{nx}, r_{ny}\}$ Bresenham measurement.

You can also see this from Fig. 3: the candidate points after a z-move are the points $\{P5, P4, P6$ and $P7\}$ corresponding to a $\{z\}$ -, $\{z,y\}$ -, $\{z,x\}$ -, $\{z,y,x\}$ -move. To select the best point, one needs six decision variables of which two are independent, which correspond with the decision variables D_x and D_y of Bresenham's dual algorithm. D_y selects $P5$ or $P4$ and D_x selects $P5$ or $P6$. Suppose D_y selects $P5$ and D_x selects $P6$, the dual Bresenham's algorithm selects $P6$, but point $P7$ can be better than $P6$. So, even when the major axis principle holds, Bresenham's dual algorithm can be wrong. The real problem is that the major axis principle often does not hold, this means that the best point of the points $\{P1, P3, P5\}$ is not $P5$. Suppose that $P3$ is the best, then the best move is selected from $\{P3, P2, P4, P7\}$. The 3D-algorithms which don't use the major axis principle have to select the best point from every possible major axis, and to select the best of the former best selected points.

For the imperfect 3D-line IPO's one applies the PRM-cs as in §1.6 using Table 1, hence the current imperfect discrete lines are very easily converted to constant speed 3D-line IPO's.

1.7.2. The global definition of a major axis discrete 3D-line [(18),[16] (§1.7.3 DAG-ipo)

Oscar Figueiredo, J.-P. Reveillès[15], and R.D. Hersch [16] recognized that the global definition is not ideal from a theoretical point of view. It has several drawbacks or limitations:

- (a) The 3rd projection is generally not the closest set of points to the 3rd Euclidean projection,
- (b) The points of the discrete line are, generally, not the set of the closest points of the given Euclidean line.

This simply means, that the discrete 3D-lines, for which the major axis principle holds, are, generally, imperfect.

We show, in §1.7.4, that from a practical viewpoint, this definition gives discrete 3D-lines whose accuracy is much less than the accuracy of the best discrete 3D-lines (e.g. 37% worse).

The Diophantine inequalities are in fact a main property. Contrary to the local Bellman principle, the Diophantine inequalities may be used as global (naive) definitions of a major axis 3D-line, i.e. for the major z-axis 3D-line:

Using Floor form:

$$0 \leq \text{Remainder}_{\text{Floor}}[F_x \mid -F_y] < 1;$$

$$-\frac{Lk}{2} \Delta \leq F_x[y_n - y_s, z_n - z_s] < \frac{Lk}{2} \Delta;$$

$$-\frac{Lk}{2} \Delta \leq -F_y[x_n - x_s, z_n - z_s] < \frac{Lk}{2} \Delta;$$

Using Ceiling form:

$$0 \leq \text{Remainder}_{\text{Ceiling}}[F_x \mid -F_y] < 1;$$

$$-\frac{Lk}{2} \Delta < F_x[y_n - y_s, z_n - z_s] \leq \frac{Lk}{2} \Delta;$$

$$-\frac{Lk}{2} \Delta < -F_y[x_n - x_s, z_n - z_s] \leq \frac{Lk}{2} \Delta;$$

With $y_n = \text{Floor}\left[\frac{L_j}{L_k}(z_n - z_s) + 0.5\right]$ and $x_n = \text{Floor}\left[\frac{L_i}{L_k}(z_n - z_s) + 0.5\right]$ or

with $y_n = \text{Ceiling}\left[\frac{L_j}{L_k}(z_n - z_s) - 0.5\right]$ and $x_n = \text{Ceiling}\left[\frac{L_i}{L_k}(z_n - z_s) - 0.5\right]$.

For 2D-lines, rounding corresponds with the minimum absolute perpendicular distance, but for 3D-lines, rounding does not correspond with the minimum absolute distance.

We also have a round up and round down problem: e.g. $\text{Floor}[3.5 + 0.5] \equiv 4$,

$\text{Ceiling}[3.5 - 0.5] \equiv 3$, $\text{Floor}[3.7 + 0.5] \equiv \text{Ceiling}[3.7 - 0.5] \equiv 4$.

So, when the decimal value of the coordinate of an intersection point equals 0.5, one must sometimes apply a correction. It is easy to find the corrections, using a Karnaugh map and a proper example, but the problem is complex because one must sometimes apply a correction when the decimal value of the coordinate of an intersection point is near 0.5.

The reason of the “equals 0.5” error is simple: the arithmetic definition of a discrete line does not use the “integrated priority”, but that is not the cause of the “near 0.5” error.

The reason for the “near 0.5” error is fundamental: 3D-distance.

For 3D-lines, the global definition does not give the MaxErr nor the MinMaxErr of the

discrete 3D-line, but we know that $\text{MinMaxErr} \leq \text{MaxErr} \leq \frac{\sqrt{2}}{2} \Delta$ for perfect and imperfect

3D-lines. For a perfect line MaxErr equals MinMaxErr, but this value is not a priori known.

The MaxErr of the line {10, 10, 10} is zero when we apply the LSD with integrated priority,

but the discrete 3D-line complies with that $0 \leq \text{MaxErr} \leq \frac{\sqrt{2}}{2} \Delta$. The imperfect discrete 3D-

line and the global definitions satisfy the worst case design and can be applied.

The 2D Diophantine inequality improves the upper bound of the maximum error MaxErr :

For discrete 2D-lines $\{L_i, L_j\}$, with $L_i > L_j > 0$, the signed error equals

$$\text{sErr}[x_n, y_n] \equiv \frac{-F_z[x_n - x_s, y_n - y_s]}{\sqrt{L_i^2 + L_j^2}} \equiv \frac{-(L_i * (y_n - y_s) - L_j * (x_n - x_s))}{\sqrt{L_i^2 + L_j^2}}, \text{ the remainders in respectively}$$

$$\text{Floor- and Ceiling form equal } \text{Remainder}_{\text{Floor}}[-F_z] = r_{zf} = -\frac{F_z[x_n - x_s, y_n - y_s]}{L_i * \Delta} + \frac{1}{2},$$

$$\text{Remainder}_{\text{Ceiling}}[F_z] = r_{zc} = \frac{F_z[x_n - x_s, y_n - y_s]}{L_i * \Delta} + \frac{1}{2}.$$

The global definition for the major x-line, in respectively Floor- and Ceiling form is

respectively $0 \leq r_{zf} < 1$ and $0 \leq r_{zc} < 1$. Using the signed error, the global definitions become

$$-\frac{L_i}{\sqrt{L_i^2 + L_j^2}} \frac{\Delta}{2} \leq \text{sErr}[x_n, y_n] < \frac{L_i}{\sqrt{L_i^2 + L_j^2}} \frac{\Delta}{2} \quad \text{or} \quad -\frac{L_i}{\sqrt{L_i^2 + L_j^2}} \frac{\Delta}{2} < \text{sErr}[x_n, y_n] \leq \frac{L_i}{\sqrt{L_i^2 + L_j^2}} \frac{\Delta}{2}.$$

This means that MaxErr of the discrete major x-axis 2D-line is less or equal than $\frac{L_i}{\sqrt{L_i^2 + L_j^2}} \frac{\Delta}{2}$

instead of the upper bound $\frac{\Delta}{2}$ of Van Aken [7]. Even in this case, MinMaxErr is not known.

1.7.3. *Current imperfect 3D-lines IPO's*

The current 3D-lines are:

1. 3D-DAG-ipo [15,18]:

Jean-Pierre Reveillès [17,16] defined an analytical definition for a discrete naïve 2D-line, and thereby, initiated the discrete geometry, which associates at the Euclidean object a discrete representation in the form of a Diophantine inequality, with has an integer solution. This was extended by Reveillès, Figueredo [15] and I. Debled-Rennesson [18] to a naïve 3D-line.

The 3D-DAG-ipo and the definition of the discrete 3D-line $\{L_i, L_j, L_k\}$ is based on

- One applies the major axis principle and the Simple DDA-algorithm [19] extended to the dependent $\{x, y\}$ -intersections with the vertical plane $z=z_n$,
- One applies Euclid's Division in Floor form or in Ceiling form,
- One applies the Floor Identities or the Ceiling Identities (3.3) of [20, pg. 68] properly to the intersection points.

You cannot use the Floor method and the Ceiling method at the same time, this means that the closed or open global definitions of the discrete 3D-line cannot be used.

The global definition is equivalent to the main property of Euclid's Division:

$$0 \leq \text{Remainder}_{\text{Floor}}[F_x] < 1 \quad \text{or} \quad 0 \leq \text{Remainder}_{\text{Ceiling}}[F_x] < 1,$$

$$0 \leq \text{Remainder}_{\text{Floor}}[-F_y] < 1 \quad \text{or} \quad 0 \leq \text{Remainder}_{\text{Ceiling}}[-F_y] < 1.$$

$$\text{Note: } \text{Remainder}_{\text{Floor}}[F_x] \equiv \frac{F_x[y_n - y_s, z_n - z_s]}{L_k \Delta} + \frac{1}{2}; \quad \text{Remainder}_{\text{Ceiling}}[F_x] \equiv -\frac{F_x[y_n - y_s, z_n - z_s]}{L_k \Delta} + \frac{1}{2}$$

The generated 3D-line is identical with the results of Bresenham's extended 3D-line.

2. Bresenham's extended 3D-line [24]:

One applies Bresenham's algorithm to the projection of the line segments in the $\{z, x\}$ and the $\{z, y\}$ plane. The major z-axis principle holds, but it is very easy to convert this algorithm to a constant speed IPO using a PRM-cs. The extended 3D-Bresenham-ipo of [25] is faster and more efficient, they agree that it is not the most precise and they show

that the maximum error MaxErr is less than equal to $\frac{\sqrt{2}}{2}$. When the production manager

agrees with that bound, the IPO can be applied successfully. The generated discrete grid points, using rounding, for the line from $\{0, 0, 0\}$ to $\{L_i, L_j, L_k\} \in \mathbb{N}^3$, $0 \leq L_i \leq L_j < L_k$ equals

$$\left\{ \text{Floor}\left[\frac{L_i}{L_k} z_n + 0.5\right], \text{Floor}\left[\frac{L_j}{L_k} z_n + 0.5\right], z_n \right\} \text{ for } z_n=0, z_n \leq L_k, z_n++.$$

3. 3D-PRM-ipo (§1.9, Case (#B) and algorithm constant speed 3D-PRM-ipo):

A PRM-ipo looks as a PRM-cs, and the 2D-PRM-ipo is given in §1.8. The PRM-ipo is the

fastest of all line IPO's, but it is also a major axis IPO. The generated 3D-line is identical with the results of Bresenham's extended 3D-line.

4. Toutant's best 3D-line [26]:

Toutant corrects the DAG-ipo and Bresenham's extended 3D-line when the decimal value of the coordinate of an intersection point equals 0.5 but the limitations (a) and (b) of §1.7.2 remain. Toutant defines his best discrete 3D-line, in our terminology, as: "the minimal 26-connected set of closest integer points". When you compare his definition with our definition of §1.3, then his "Minimal" is wrongly defined but it certainly means that the major axis principle holds, and that is wrong, too.

The papers [27,28] of C.E. KIM of respectively of 1981 and 1983: "two projections along the major axis of the 3D-line produce two 2D-lines and these suffice to compute the 3D-line", caused that for a discrete 3D-line the major axis principle must hold. That is right for the projections along any axis ($L \cdot F_n \equiv 0$, $L \cdot p_n \equiv 0$), as well for the continuous 3D-line as for the discrete 3D-line. The trouble, when you want to apply the projections along a minor axis, is that you don't know the right value for the component of that minor axis. So, the rule of the major axis principle simplifies the design of IPO's and the global definition of the discrete 3D-line is just a property of that design and it is much easier to start the design with the global definition of the discrete lines and extend that definition to discrete objects, without saying that the major axis principle must hold.

1.7.4 Discussion of the major axis 3D-ipo's for the major z-axis $L=\{Li,Lj,Lk\}=\{49,74,82\}$ 3D-line

Initially, we wanted to treat all lines in this part, but that is impossible, therefore, we split the paper into three parts, the 2nd part will discuss the IPO's of the perfect 3D-lines and the 3rd part will treat the QSICs and 3D-NURBS.

Point Error	Points of the 3D-line $L=\{49,74,82\}$ $\{x_{Ln}, y_{Ln}\} = \left\{ \frac{49}{82} * z_n, \frac{74}{82} * z_n \right\}$			Rounding	LSD
$\{x_{Ln}, y_{Ln}, z_n\}$	$\{9.56, 14.44, 16\}$	$\{15.54, 23.46, 26\}$	$\{21.51, 32.49, 36\}$		
Rounded $P_n^{\uparrow}$	$\{10, \mathbf{14}, 16\}$	$\{16, \mathbf{23}, 26\}$	$\{22, \mathbf{32}, 36\}$	Yes	No
Error[$P_n^{\uparrow}$]	0.614194	0.648316	0.682438	"	"
LSD P_n	$\{10, \mathbf{15}, 16\}$	$\{16, \mathbf{24}, 26\}$	$\{22, \mathbf{33}, 36\}$	No	Yes
Error[P_n]	0.485175	0.485669	0.488551	"	"
$\{x_n^{\uparrow} - x_{Ln}, y_n^{\uparrow} - y_{Ln}\}$	$\{+, -\}$	$\{+, -\}$	$\{+, -\}$	$\{x_n, y_n\} = \{x_n^{\uparrow}, y_n^{\uparrow}\}$	$\{x_n, y_n\} = \{x_n^{\uparrow}, y_n^{\uparrow} + 1\}$

Table 2

The Error[P_n] equals $\text{Abs} \left[\frac{\text{Norm}[\text{Cross}[L, P_n]]}{\text{Norm}[L]} \right]$ when $P_s = \{0,0,0\}$.

We convert the IPO's of §1.7.3 using the PRM-cs using $\{LM_{cs}, LN_{cs}\}$ of Table 1 and the initial AC_{cs} equals $LN_{cs}/2$. The values of $\{\text{MaxErr}, \text{Cost}, \text{npuls}\}$ equal $\{0.682438, 10.8273, 121\}$.

The values of {MaxErr, Cost, npuls} of the discrete line of the best IPO equal {0.497924, 10.0968, 121}. The simulated length of the four imperfect IPO's equals npuls and that equals the rounded length of the line: (Round[Norm[L]]=Round[120.835]), hence the four IPO's generate constant speed discrete lines when one adds a proper PRM-cs (Table 1). The Cost of the imperfect IPO's of §1.7.3 increases with 7%, but the MaxErr increases with 37.1 %.

Paths	$P_s=P_n$	3D-Line {49,74,82}		$P_e=P_n$	Bellman's principle	Major axis principle	Best path
Path A1	{34,51,57}	{34,52,57}	{35,52,58}	{35,53,59}	OK	Wrong	A1
A1: $ \rho_n $	0.332064	0.465581	0.477707	0.24689	"	"	
Path B1	{34,51,57}	$\Rightarrow$	{35,52,58}	{35,53,59}	Wrong	OK	B1
B1: $ \rho_n $	0.332064	$\Rightarrow$	0.477707	0.24689	"	"	
Path A2	{03,04,05}	{03,05,05}	{04,06,06}	{04,06,07}	OK	Wrong	A2
A2: $ \rho_n $	0.408877	0.381943	0.487077	0.248962	"	"	
Path B2	{03,04,05}	$\Rightarrow$	{04,06,06}	{04,06,07}	Wrong	Not-conn.	B2
B2: $ \rho_n $	0.408877	$\Rightarrow$	0.487077	0.248962	"	"	

Table 3

The causes of imperfect discrete 3D-lines:

1. 3D Rounding is not equivalent with 3D LSD

The rounded grid points of a 2D-line are always the LSD grid points, but the rounded grid points of a 3D-line are, generally, not the LSD grid points, and that is the cause of the limitation (a) of §1.7.2.

Define RoundUp as $u_n^{\uparrow} \equiv \text{Floor}[u_{Ln}+0.5]$ and RoundDown as $u_n^{\downarrow} \equiv \text{Ceiling}[u_{Ln}-0.5]$.

When the decimal value of u_{Ln} is different from 0.5, then RoundUp $\equiv$ RoundDown.

The rounded points, in Floor form, for the major z-axis line {49,74,82} are

$$P_n^{\uparrow} = \{x_n^{\uparrow}, y_n^{\uparrow}, z_n\} = \left\{ \text{Floor}\left[\frac{49}{82} * z_n + \frac{1}{2}\right], \text{Floor}\left[\frac{74}{82} * z_n + \frac{1}{2}\right], z_n \right\}.$$

3D lines have many rounded points which are not identical with the LSD points, even when the major z-axis principle holds and when RoundUp $\equiv$ RoundDown. Hence limitation (a) is not a pure 0.5 rounding problem. The rounded points are also different

from the LSD points when the decimal value of the coordinate of $\frac{49}{82} * z_n$ or $\frac{74}{82} * z_n$ is near

0.5.

As you see from Table 2, the simple empirical corrections for the Table 2 are $x_n = x_n^{\uparrow}$

when $x_n^{\uparrow} - x_{Ln} > 0$ and $y_n = y_n^{\uparrow} + 1$ when $y_n^{\uparrow} - y_{Ln} < 0$.

Toutant's 3D-line corrects this error only when the decimal values equal to 0.5.

2. Infrequently, the dual of Bellman's principle of optimality does not hold for 3D-lines

The LSD criterium applies, locally, the dual of Bellman's principle of optimality, this means that, when locally, the closest points are selected, then one obtains also, globally, the minimum Cost. We know that the dual of Bellman's principle of optimality does not hold for 3D-6-connected curves [5, pp. 27], but, occasionally, it is also not valid for a 26-connected 3D-line. The path A1 and B1 of Table 3 have the same MaxErr (0.477707), but the Cost of path B1 is less than the Cost of path A1, hence path B1 is the best path, although Bellman's principle is not valid for path B1.

3. The major axis principle, generally, does not hold for 26-connected 3D-lines or curves

The major axis principle holds for 8-connected LSD-ipo's, but, generally, not for 26-connected LSD-ipo's, and that is not an exception, but it applies to most 3D-lines.

For the lines $\{49, 74, 74+i\}$ with i from 1 to 100, 92% of the lines had at least one and at most 5 possible major-z axis errors; for the lines with i from 101 to 200, 22% of the lines had at least one possible major-z axis error.

The minimum MaxErr of the "best" 3D-line $\{L_i, L_j, L_k\} = \{49, 74, 82\}$ equals 0.497924. The major axis principle holds for the four IPO's of §1.7.3 and their MaxErr equals 0.682438, hence the accuracy is 37% worse.

The path A2 and B2 of Table 3 have the same MaxErr (0.487077), but the cost of B2 is smaller than the cost of A2, but path B2 is not 26-connected as the y-move from $\{03, 04, 05\}$ to $\{04, 06, 06\}$ is impossible. The major axis principle for path A2 is not valid, nevertheless, it is the best path.

1.8 Initializations & Major axis 2D-line IPO's: Simple-DDA-, DAG- and PRM-ipo[21]

We define the Euclidean 2D-line $\{L_i, L_j\}$ as $F_p \equiv F_z[x_p, y_p] \equiv L_i * (y_p - y_s) - L_j * (x_p - x_s)$, and $F_A[0]$ as $F_A[0] \equiv F_z[0, 0] \equiv -(L_i * y_s - L_j * x_s)$. We defined $S_{Lxy} = S_{LEFT} * S_x * S_y$ [5, (6)] with

$\{S_x, S_y\} \equiv \text{Sign}[\{L_i, L_j\}]$ and $S_{LEFT} \equiv -1$ as this line is always positive at the left side, hence

$S_{Lxy} \equiv -S_x * S_y$. Therefore, S_{Lxy} equals -1 in the 1st and 3rd quadrant and +1 in the 2nd and 4th quadrant.

We want to show that the initial AC_{ipo} of the PRM-ipo $\{L_i, L_j\}$ equals

$$AC_{ipo}[0] \equiv \frac{LN_{ipo}}{2} - d_M * S_{Lxy} * F_A[0] \text{ and the initial } AC_{cs} \text{ of the PRM-cs equals } AC_{cs}[0] \equiv \frac{LN_{cs}}{2}.$$

We define the variables of the PRM-cs $\{L_i, L_j\}$ (Table 1) as $LM_{cs} = \text{Max}[\text{Abs}[\{L_i, L_j\}]]$ and

$$LN_{cs} \equiv LS \equiv \text{Norm}[\{L_i, L_j\}] \equiv \sqrt{L_i^2 + L_j^2}.$$

We define the variables of PRM-ipo $\{L_i, L_j\}$ as $LM_{ipo} = \text{Min}[\text{Abs}[\{L_i, L_j\}]]$ and

$$LN_{ipo} = \text{Max}[\text{Abs}[\{L_i, L_j\}]].$$

The core of the PRM-ipo is

Major x-axis PRM-ipo

...

$x_n = x_n + S_x$; $AC_{ipo} = AC_{ipo} - LM_{ipo}$;

If $[AC_{ipo} > 0, \text{Goto}[\text{LIST}]]$;

$y_n = y_n + S_y$; $AC_{ipo} = AC_{ipo} + LN_{ipo}$;

Label[LIST];

Major y-axis PRM-ipo

...

$y_n = y_n + S_y$; $AC_{ipo} = AC_{ipo} - LM_{ipo}$;

If $[AC_{ipo} > 0, \text{Goto}[\text{LIST}]]$;

$x_n = x_n + S_x$; $AC_{ipo} = AC_{ipo} + LN_{ipo}$;

Label[LIST];

...

...

Conditions: $\frac{LM_{ipo}}{LN_{ipo}} \leq 1$, $LM_{ipo} > 0$, $LN_{ipo} > 0$, $AC_{ipo} > 0$ [21].

Similar to the three first 3D-IPO's of §1.7.3, we also have three major axis IPO's but that poses no problems as 2D-rounding for 2D-lines is equivalent to the LSD criterium. Mostly, one does not apply the "priority principle" and 2D-IPO's which do not apply this principle will be called nearly-perfect IPO's. So, the nearly-perfect IPO's are major axis 2D IPO's, the major axis principle holds, one applies the LSD-criterium (or Rounding), but not the priority principle.

These nearly-perfect 2D-IPO's are:

1. 2D-DAG-ipo's.
2. 2D-Bresenham-ipo's.
3. 2D-PRM-ipo's.

The 2D-BRES-ipo is a perfect IPO, the major axis is not, a priori, used, therefore the monotonic segment must not be subdivided. These four IPO's must, intrinsically, use the same decision variable; therefore, we naturally take the decision variable AC_{ipo} of the PRM-ipo and the decision variable d_H or d_V of the BRES-ipo. In §1.5, we defined the three decision variables for the BRES-ipo using the polar line of a conic, and as the polar of a line is identical with the line, these decision variables are also valid for the 2D-line F_p . The decision variables of the 2D-BRES-ipo are $\{d_M, d_H, d_V\} \equiv \text{Sign}[\{\rho_C^2 - \rho_B^2, \rho_D^2 - \rho_B^2, \rho_C^2 - \rho_D^2\}] \equiv -S_{Lxy} * \text{Sign}[\{F_M, F_H, F_V\}]$.

We consider two cases:

❖ Case1: The major x-axis holds.

Then $\frac{LN_{ipo}}{\Delta} \equiv |L_i| > |L_j| \equiv \frac{LM_{ipo}}{\Delta}$, $\rho_C^2 - \rho_B^2 \equiv -S_{Lxy} * F_M > 0$, $d_M \equiv 1$,

$\rho_D^2 - \rho_B^2 \equiv -S_{Lxy} * F_H \equiv -S_{Lxy} * F_A + \frac{|L_i|}{2} \Delta - |L_j| \Delta$ corresponds with selects P_D when $\rho_D^2 - \rho_B^2 \leq 0$

or when $-S_{Lxy} * F_H \leq 0$. The PRM-ipo select P_D when $AC_{ipo} \leq 0$, but the value at that moment equals $AC_{ipo} - LM_{ipo}$, hence $AC_{ipo} - LM_{ipo} \equiv -S_{Lxy} * F_H$, or

$$AC_{ipo} \equiv LM_{ipo} - S_{Lxy} * F_H \equiv LM_{ipo} - S_{Lxy} * F_A + \frac{|L_i|}{2} \Delta - |L_j| \Delta.$$

As $LM_{ipo} \equiv |L_j| \Delta$ and $LN_{ipo} \equiv |L_i| \Delta$, we get $AC_{ipo} \equiv \frac{LN_{ipo}}{2} - S_{Lxy} * F_A$ and initial

$$AC_{ipo}[0] \equiv \frac{LN_{ipo}}{2} - S_{Lxy} * F_A[0]$$

$$AC_{ipo}[0] \equiv \frac{LN_{ipo}}{2} - d_M * S_{Lxy} * F_A[0]$$

❖ Case2: The major y-axis holds.

Then $\frac{LM_{ipo}}{\Delta} \equiv |L_i| < |L_j| \equiv \frac{LN_{ipo}}{\Delta}$, $\rho_C^2 - \rho_B^2 \equiv -S_{Lxy} * F_M < 0$, $d_M \equiv -1$,

$\rho_C^2 - \rho_D^2 \equiv -S_{Lxy} * F_V \equiv -S_{Lxy} * F_A + |L_i| \Delta - \frac{|L_j|}{2} \Delta$ corresponds with select P_D when $\rho_C^2 - \rho_D^2 \geq 0$

or when $-S_{Lxy} * F_V \geq 0$ or when $S_{Lxy} * F_V \leq 0$. The PRM-ipo selects P_D when $AC_{ipo} \leq 0$, but the value at that moment equals $AC_{ipo} - LM_{ipo}$, hence $AC_{ipo} - LM_{ipo} \equiv S_{Lxy} * F_V$, or

$$AC_{ipo} \equiv LM_{ipo} + S_{Lxy} * F_H \equiv LM_{ipo} + S_{Lxy} * F_A - |L_i| \Delta + \frac{|L_j|}{2} \Delta.$$

As $LM_{ipo} \equiv |L_i| \Delta$ and $LN_{ipo} \equiv |L_j| \Delta$, we get $AC_{ipo} \equiv \frac{LN_{ipo}}{2} + S_{Lxy} * F_A$ and initial

$$AC_{ipo}[0] \equiv \frac{LN_{ipo}}{2} + S_{Lxy} * F_A[0]$$

- For both cases $AC_{ipo}[0] \equiv \frac{LN_{ipo}}{2} - d_M * S_{Lxy} * F_A[0]$.
- Most of the time $F_A[0]$ is zero, but the major x-axis PRM-ipo is always different from the major y-axis PRM-ipo. Therefore, a quarter circle cannot be realized with one PRM-ipo.
- All published PRM-ipo's or comparable IPO's use a wrong initialization.

The initialization of the PRM-cs:

We suppose that $0 < L_j < L_i$, then $nkey \equiv \frac{LM_{cs}}{LN_{cs}} * npuls \equiv \frac{L_i}{LS} * npuls$ and for the PRM-ipo $\{L_i, L_j\}$

$$y_n = \frac{L_j}{L_i} * x_n \text{ discretizes the line } \frac{x - x_s}{L_i} \equiv \frac{y - y_s}{L_j}.$$

We can relate the PRM-cs with a modified PRM-ipo:

Instead of the line $\frac{x - x_s}{L_i} \equiv \frac{y - y_s}{L_j}$, we could take the line $\frac{x - x_s}{L_i} \equiv \frac{a - a_s}{LS}$, and that line is

$F_a = LS * (x_n - x_s) - L_i * (a_n - a_s)$ with $F_a[0] = -(LS * x_s - L_i * a_s)$ and a_n is the arclength or the length of the line. Suppose that we always start with $\{x_s, a_s\}$ equal to $\{0, 0\}$ then $F_a[0] \equiv 0$

Now, for $a_n \equiv npuls$ and $nkey \equiv x_n$ we get the PRM-cs, and the initial npuls and nkey are always equal to 0, therefore the initial value of AC_{cs} equals $AC_{ipo}[0] \equiv \frac{LN_{cs}}{2}$.

Properties of the PRM-ipo and the PRM-cs:

- For case1, AC_{ipo} corresponds with the residue in the point P_h , and for case2, AC_{ipo} corresponds with the residue in the point P_v (Fig. 4). Therefore, the PRM-ipo is, in essence, a midpoint algorithm.
- The simulated npuls of the PRM-cs equals, in real-time, the rounded length of the generated discrete line,
- The PRM-ipo is a major axis IPO, and the PRM-cs expects that the cascaded IPO is a major axis IPO,
- The tangential speed of the cascaded IPO is the reverse of the pulse rate $\frac{nkey}{npuls}$ of the PRM-cs,
- A PRM-cs applies to 2D- and 3D-IPO's (Circles, Conics, 3D-lines, QSICS, LSD-NURBS),

- In general, a PRM-ipo cannot be applied successfully, to Circles, Conics, because the monotonic segment must be subdivided into major axis segments,
- The PRM-cs is a pulse-rate-multiplier which is independent of the major axis of the line or the curve. Therefore, the PRM-cs is valid for the complete monotonic segment, but when L_i and L_j changes, the changes of the tangent must be communicated to the PRM-cs.

Errors of all published PRM-ipo's:

- All published PRM-ipo's or comparable IPO's use a wrong initialization.
- The MaxErr depends on the initialization of AC_{ipo} , e.g. for the line {4,3}, the MaxErr increases with 50% and the Cost with 130% when the initialization of $AC_{ipo}[0]$ becomes $AC_{ipo}[0] = 0$ or $AC_{ipo}[0] = LN_{ipo}=4$, but the constant speed is independent of the initialization of the PRM-ipo.

Possible errors of a PRM-cs:

- The ratio $\frac{LS}{npuls}$ depends on the initialization of AC_{cs} , e.g. for the line {4,3}, that ratio increases with 25% for $AC_{cs}[0]=0$, and the ratio decreases with 16.7% for $AC_{cs}[0]=LN_{cs}=5$, but the MaxErr and the Cost are independent of the initialization of the PRM-cs.
- PRM-cs is, as far as we know, completely new. You have to be aware that the PRM-cs looks like a PRM-ipo, this means that the PRM-cs expects a major axis IPO, that does not mean that the PRM-cs, itself, must be changed; that will be explained in Part II.

1.9 Constant Speed BRM-ipo or Software DDA-IPO

The Symmetric DDA-algorithm [19, pp. 70-71] is a constant speed IPO, not based on the major axis principle. The increments, for a 2D-line, are $\frac{\Delta x}{L_i} = \frac{\Delta y}{L_j} = \frac{\Delta a}{LS}$ and the incremental

equations are $x[n] = x[n-1] + \Delta x$ and $y[n] = y[n-1] + \Delta y$. The calculation error of the increments accumulates, and that is one of the main reasons that the symmetric-DDA-IPO's became obsolete. But we can write, the symmetric-DDA, directly as $x[n] = x_s + \frac{L_i}{LS} * s[n]$,

$y[n] = y_s + \frac{L_j}{LS} * s[n]$, with $s[n] = a[n] - a[0] \approx npuls$. We can create npuls very easily with a PRM-

cs and the integer coordinates become $x_n = x_s + \text{Floor}[\frac{L_i}{LS} * npuls + 0.5]$,

$y_n = y_s + \text{Floor}[\frac{L_j}{LS} * npuls + 0.5]$. We can faster implement the Floor function with a PRM-ipo

and delete the PRM-cs, except the increment $npuls=npuls+1$. We call this implementation with two PRM-ipo's in cascade and where npuls functions as the major axis, the BRM-ipo. Danielson [22, Fig.4, 5] called it the "Binary Rate Multiplier", Spiros Papaioannou [23] called it the "linear DDA IPO with speed control", he also used a "pulse rate multiplier", but John Peatman [21, pp. 350-353] defines the PRM-ipo much clearer. Koren [9],[10] called the BRM-ipo a "software IPO" and [11, pp. 79, Fig. 3.7] calls it a "software DDA IPO". The Fig. 3.7

$\{L, Q1, A, Q2, B\}$ of [11] resembles the BRM-ipo: $A == |L_i|$, $B == |L_j|$, $L == LS$, the decision variables of their Q1-ipo and our AC_x -ipo are identical, hence $-(Q1 + S_x * L_i - LS) == AC_x - S_x * L_i ==> Q1 == LS - AC_x$ and analogous $Q2 == LS - AC_y$. They initialize Q1 and Q2 as zero and that corresponds with the wrong initialization of AC_x and AC_y as LS.

When we convert their decision variables to our decision variable, then the published BRM-ipo's used LS (sometimes zero) as initialization, in our AC_{ipo} terminology. So, all published BRM-ipo's or software DDA's use the wrong initialization and their results were not good. They applied the linear BRM-ipo's to circle BRM-ipo's, and the results got worse.

The best linear BRM-ipo uses the proved initialization, but even in that case it does not make sense to use a BRM-ipo, it can better be replaced with a PRM-cs in cascade with a PRM-ipo, and the circle BRM-ipo must be replaced with a PRM-cs in cascade with a circle BRES-ipo (Table 6 (#A) versus (#B) and 1.10.2).

In many publications the word DDA is misused, [29] even calls Bresenham's 2D-line IPO a DDA-algorithm. We don't further discuss the non-major axis DDA-algorithms, but there exist, many, even well-known and referenced, non-constant speed, major axis DDA-ipo's, derived from the Simple DDA-ipo [19].

Possible errors of a linear 2D-BRM-ipo and 3D-BRM-ipo:

Results of BRM-ipo's for the line $\{L_i, L_j\} = \{4, 3\}$, $LS = \text{Norm}[\{L_i, L_j\}] = 5$, $nkey == 4$					
(#)	MaxErr	npuls	Cost	Results:	When:
(1)	0.4	5	0.4	MaxErr: OK, Cost: Wrong, npuls: OK	$\text{Floor}\{((L_i/LS)*npuls + .5, (L_j/LS)*npuls + .5)\}$
(2)	0.4	5	0.4	MaxErr: OK, Cost: Wrong, npuls: OK	$AC_x = AC_y = LS/2$
(3)	0.4	5	0.24	MaxErr: OK, Cost: OK, npuls: Wrong	$AC_x = AC_y = 0$
(4)	0.4	4	0.24	MaxErr: OK, Cost: OK, npuls: OK	$AC_x = AC_y = LS$
(5)	0.4	5	0.24	MaxErr: OK, Cost: OK, npuls: OK	$Q1 = Q2 = 0$

Table 4

Results of BRM-ipo's for the line $\{L_i, L_j\} = \{7, 5\}$, $LS = \text{Norm}[\{L_i, L_j\}] = 8.6$, $nkey == 7$					
(#)	MaxErr	npuls	Cost	Results:	When:
(1)	0.465	8	0.689	MaxErr: OK, Cost: OK, npuls: ~ OK	$\text{Floor}\{((L_i/LS)*npuls + .5, (L_j/LS)*npuls + .5)\}$
(2)	0.465	8	0.689	MaxErr: OK, Cost: OK, npuls: ~ OK	$AC_x = AC_y = LS/2$
(3)	0.58	8	0.973	MaxErr: Wrong, Cost: Wrong, npuls: ~ OK	$AC_x = AC_y = 0$
(4)	0.697	9	0.865	MaxErr: Wrong, Cost: Wrong, npuls: OK	$AC_x = AC_y = LS$
(5)	0.697	9	0.865	MaxErr: Wrong, Cost: Wrong, npuls: OK	$Q1 = Q2 = 0$

Table 5

Cases (#1) and (#2), and cases (#4) and (#5) in Table 4 and Table 5 are identical. Case (#5) is the software DDA IPO of [11] with initial Q1 and Q2 equal to zero and the equivalent BRM-ipo with initial AC_x and AC_y equal to LS. Case (#1) calculates $\{x_n, y_n\}$ as

$\text{Floor}\left[\left\{\frac{L_i}{LS} * npuls + 0.5, \frac{L_j}{LS} * npuls + 0.5\right\}\right]$, (#2) uses the 2D-BRM-ipo with the initials

$AC_x = AC_y = \frac{LS}{2}$, and using the equivalent relations of §1.6.1, npuls equals

$\text{Ceiling}[\frac{LS}{\text{Max}[L_i, L_j]} * nkey - 0.5]$ or $\text{Ceiling}[\frac{LS}{\text{Max}[L_i, L_j]} * (nkey - 0.5)]$. That gives, for $nkey \equiv 7$,

{9,8} and for $nkey \equiv 4$, {5,5}. This means that the value 8 or 9 of npuls, can be considered as a good result for Table 5 and that the value 5 of npuls is a good result for Table 4.

The general conclusion is that the 2D-BRM-ipo with $AC_x = AC_y = \frac{LS}{2}$ is the best, but as the

principle of priority cannot be applied, the 2D-BRM-ipo is not a perfect 2D-BRM-ipo, therefore the Cost is not always the best, but the MaxErr is OK. To my knowledge, we do not know any paper which discusses the initial value of a 2D-BRM-ipo.

Table 6 shows the result of three 3D-line constant speed IPO's. The feedrate is constant as npuls, equal to Round[LS] shows. Item (#A) corresponds with a well-designed 3D-BRM, nevertheless the Cost is bad and the MaxErr is 46.8 larger than the minimum MaxErr. Item (#B) corresponds with the 3D-PRM of §1.7.3, and although it contains one PRM-cs and two 2D-PRM-ipo's, it is better than the 3D-BRM-ipo, which contains three 2D-PRM-ipo's. This, once again, shows that the BRM-ipo must be replaced by a PRM-cs in cascade with an actual non-constant speed IPO, and preferably the BRES-ipo. Item (#C) corresponds with the Best 3D-line BRES-ipo.

Results for the line $\{L_i, L_j, L_k\} = \{49, 74, 82\}$, $LS = \text{Norm}[\{L_i, L_j, L_k\}] = 120.835$, $nkey \equiv 82$					
(#)	MaxErr	npuls	Cost	Results:	When:
(A)	0.73099 $> \frac{\sqrt{2}}{2}$!!!	121	19.8654	MaxErr: Wrong, Cost: Wrong	3D BRM-ipo $AC_x = AC_y = AC_z = LS/2$
(B)	0.682438	121	10.8273	MaxErr: Wrong, Cost: Wrong	PRM-cs + 3D Major z-axis PRM-ipo $AC_x = AC_y = L_k/2$
(C)	0.497924	121	10.0968	MaxErr: OK, Cost: OK	Best 3D-BRES-ipo
(D)	0.497924	121	10.0968	Interrupt==121	listSIM-ipo (§1.10)

Table 6

Finally, we will detail, a little, the 3D-BRM-ipo and the constant speed 3D-PRM-ipo:

3D-line $L = \{ L_i, L_j, L_k \}$, $\{ LM_x, LM_y, LM_z \} = \text{Abs}[L]$, $\{ S_x, S_y, S_z \} = \text{Sign}[L]$,
 $LS = \text{Norm}[L]$, $LM_{cs} == LM_z == \text{Max}[\{LM_x, LM_y, LM_z\}]$, (major z-axis, hence $nkey == z_n$)
 (#A) 3D-BRM-ipo (#B) constant speed 3D-PRM-ipo

Initialization:

$$AC_x = AC_y = AC_z = \frac{LS}{2}$$

$$x_n = y_n = z_n = npuls = 0$$

$$LM_{cs} = LM_z; LN_{cs} = LS;$$

$$AC_{cs} = \frac{LS}{2}; AC_x = AC_y = \frac{LM_{cs}}{2};$$

$$x_n = y_n = z_n = npuls = nkey = 0$$

Algorithm:

```
While [ $x_n \neq L_i \mid \mid y_n \neq L_j \mid \mid z_n \neq L_k$ ],
  Label[BRM];
  npuls++;  $AC_z = AC_z - LM_z$ ;
   $AC_x = AC_x - LM_x$ ;
   $AC_y = AC_y - LM_y$ ;
  Label[TESTx];
  If [ $AC_x > 0$ , Goto[TESTy]];
   $x_n = x_n + S_x$ ;  $AC_x = AC_x + LS$ ;
  Label[TESTy];
  If [ $AC_y > 0$ , Goto[TESTz]];
   $y_n = y_n + S_y$ ;  $AC_y = AC_y + LS$ ;
  Label[TESTz];
  If [ $AC_z > 0$ , Goto[LIST]];
   $z_n = z_n + S_z$ ;  $AC_z = AC_z + LS$ ;
  Label[LIST];
]
```

```
While [ $x_n \neq L_i \mid \mid y_n \neq L_j \mid \mid z_n \neq L_k$ ],
  Label[PRM-cs];
  npuls++;  $AC_{cs} = AC_{cs} - LM_{cs}$ ;
  If [ $AC_{cs} > 0$ , Goto[PRM-cs]];
   $z_n = z_n + S_z$ ;  $AC_{cs} = AC_{cs} + LS$ ;
  Label[PRM-ipo];
   $AC_x = AC_x - LM_x$ ;
   $AC_y = AC_y - LM_y$ ;
  Label[TESTx];
  If [ $AC_x > 0$ , Goto[TESTy]];
   $x_n = x_n + S_x$ ;  $AC_x = AC_x + LM_x$ ;
  Label[TESTy];
  If [ $AC_y > 0$ , Goto[LIST]];
   $y_n = y_n + S_y$ ;  $AC_y = AC_y + LM_y$ ;
  Label[LIST];
]
```

The non-orthogonal IPO of [39, Fig. 4. B] is a non-constant speed reference pulse IPO. The example Fig. 4. B shows that the MaxErr equals one, and that it is greater than the upper bound 0.5, therefore we do not consider these IPO's. The results of a PRM-cs plus the BRES-

ipo, for the ellipse $\frac{x^2}{12^2} + \frac{y^2}{18^2} \equiv 1$ from $\{18, 0\}$ to $\{0, 12\}$, are $\{\text{MaxErr}, npuls, nkey\} \equiv \{$

0.462973, 24, 22}. The number of steps for the complete ellipse equals 88 too. The

calculated total length is $\text{Arclength}[\text{ImplicitRegion}[\frac{x^2}{12^2} + \frac{y^2}{18^2} = 1, \{x, y\}]] \equiv 95.1926$ and

the rounded length of the quarter ellipse equals npuls.

1.10 Real-Time flowchart of constant-speed 2D-IPO with PRM-cs (Fig. 6)

The PRM-cs waits until an interrupt arrives. In the interrupt loop, AC_{cs} is decremented with LM_{cs} , once per interrupt. When AC_{cs} becomes less or equal to zero then LN_{cs} is added to AC_{cs} , and one continues the BRES-ipo or the near-perfect 2D-ipo (§1.8). In the latter case, the major LM_{cs} -axis principle holds, and $nkey$ becomes the coordinate of the major axis. The LIST-routine outputs the discrete coordinates and returns to the wait-routine with a positive AC_{cs} or the IPO ends.

The npuls can be deleted for a real-time system.

We can also delete the interrupt system and simulate the real-time system with a VIRT-ipo instead of the BRES-ipo. In that case we create, in LIST, the array $\text{listSIM} = \{ \{x_n, y_n, npuls\}, \}$ and this array can be executed in real-time, and we call that IPO the listSIM-ipo.

Fig. 6

$$1.10.1 \text{ The listSIM-ipo, given the array listSIM} \equiv \left\{ \overbrace{\{x_n, y_n, z_n, npuls\}}^{nl=1}, \dots, \overbrace{\{x_n, y_n, z_n, npuls\}}^{nl=nkey/M} \right\}$$

The listSIM-ipo is the fastest way to draw a curve or a NURB, but when you change the form, then you have to recreate a new listSIM. The form can often change, for example, when the radius of the tool changes, or when you have to draw the offset or parallel curves.

The algorithm is simple: we count the pulse number (ninterrupt) and when it equals npuls, we output the corresponding $\{x_n, y_n, z_n\}$. The real-time listSIM-ipo is interrupt driven.

Suppose we use vectored interrupts, then you use two vectored interrupts, and you have to set which one is active (only one of the two is active). Most of the time, the interrupt routines have nothing to do, eventually, you can then do something others, for example further fill the buffer from the file listSIM.

So, using the VIRT-ipo has many advantages:

- The VIRT algorithm is inherently very slow, clear and simple. Hence, using the VIRT-ipo to explain a complex IPO is sometimes the only possibility.
- The benchmark of all IPO's, regarding the execution time, is the listSIM-ipo. The execution time of the listSIM-ipo (#D) and the 3D-PRM-ipo (#B) are quasi the same, but the execution time of (#C), using a VIRT-ipo is 22.5 times larger or 2150 in terms of percentage. Hence, the 3D-PRM-ipo is an efficient imperfect 3D-line IPO.

- The speed is proportional to $\frac{npuls}{ninterrupt}$ and for a 3D-line $|npuls - ninterrupt| \leq 1$. So, the speed is, instantaneously, constant.

- The ratio TRB of $\frac{\text{Execution Time of listSIM-ipo}}{\text{\# of interrupts}}$ is the reference-pulse benchmark and

TRB is less than the minimal interrupt times T_c (maximum speed). The maximum time TNP to find a new gridpoint must be lower than T_c . Hence, the IPO can be used in a reference-pulse system when $TRB \leq TNP < T_c$.

- The VIR-ipo gives the additional advantage that the paper is easily reproducible. The user can simulate and test the IPO's, although, when the simulation time is limited, with a much lower resolution.

The simulation of the interrupt driven Benchmark-ipo or listSIM-ipo:

Initialization:

```
nkeyM=Length[listSIM]
Pn={xn,yn,zn}=Ps={xs,ys,zs}={0,0,0}≡ Start Point
Pe={xe,ye,ze}≡ End Point
nl=nkey=npuls=0
ninterrupt=-1
```

Algorithm:

```
While[ xn ≠ xe || yn ≠ ye || zn ≠ ze ,
  Label[StartINT1];
  ninterrupt=niinterrupt + 1;
  nl = nkey + 1; npuls = listSIM[ [ nl,4 ] ];
  Label[StopINT1];
  Label[StartINT2];
  While[ niinterrupt < npuls, niinterrupt = niinterrupt + 1 ];
  Label[StopINT2];
  {xn, yn, zn} = { listSIM[ [ nl,1 ] ], listSIM[ [ nl,2 ] ], listSIM[ [ nl,3 ] ] };
  Label[LIST];
  Output[ {xn, yn, zn} ];
  nkey = nkey + 1;
]
```

The ratio $\frac{npuls}{niinterrupt}$ is proportional with the instantaneous speed, and this ratio is quasi equal to one.

The listSIM can be input to a smart look-ahead or to a FPGA based G code controller [30] which accepts { x_n[k], y_n[k], z_n[k], npuls[k] } as input.

1.10.2 The instantaneous speed ratio $\frac{npuls}{niinterrupt}$ for a constant speed Circle equals one

To simplify the IPO's, we create the 1st quarter circle counterclockwise.

Initial:

{S_x, S_y}={-1, 1}, Ps={x_s,y_s}={R,0}, Pe={x_e, y_e}={0,R}, Pn={x_n, y_n}=Ps,

listSIMBC=listSIMVC={}, d_M= $\frac{x_n^2 + y_n^2 - R^2}{2} + \frac{y_n - x_n}{2}$, (see §1.5),

The constant speed Circle-ipo is identical with the algorithm of §1.6, when we replace

- The LSD-ipo with Bresenham's midpoint-ipo with integrated priority:
If[d_M<0, (* with d_{M0} ≡ d_M, d_V = d_{M0} + y_n/2 and d_H = d_{M0} - x_n/2 *)
{y_n = y_n + 1; d_M=d_M + y_n, If[d_M - y_n/2 ≥ 0, {x_n = x_n - 1, d_M=d_M - x_n}]},
{x_n = x_n - 1; d_M=d_M - x_n, If[d_M + x_n/2 ≤ 0, {y_n = y_n + 1, d_M=d_M + y_n}]}];
listSIMBC=Join[listSIMBC,{ { x_n, y_n, npuls } }];
- The LSD-ipo with the VIRT-ipo:
P_n= {x_n,y_n}=PminD[P_n, S_x, S_y];
listSIMVC=Join[listSIMVC,{ { x_n, y_n, npuls } }];

The virtual PminD[] subroutine :

The virtual PminD[] subroutine selects the LSD candidate point with integrated priority.

$dErrCV[P_n] := N[Abs[Dot[P_n, P_n] - R^2]]$

$Pr[P_n, P_A] := Dot[P_n - P_A, P_n - P_A]$

$PminD[P_A, Sx, Sy] := Module[\{ Pa = P_A, P_1, P_2, P_3, listPNT, listdC, listEQdC, spanEQdC, listEQ, assocSQL, PMIN1 \},$

$P_1 = Pa + \{Sx, 0\};$

$P_2 = Pa + \{Sx, Sy\};$

$P_3 = Pa + \{0, Sy\};$

$listPNT = \{P_1, P_2, P_3\};$

$listPNT = SortBy[listPNT, dErrCV[\#] \&];$

$listdC = Map[dErrCV, listPNT];$

$listEQdC = Nearest[listdC, First[listdC]];$

$spanEQdC = 1 ; ; Length[listEQdC];$

$listEQ = listPNT[[spanEQdC]];$

$assocSQL = Sort[listEQ, Pr[\#1, Pa] > Pr[\#1, Pa] \&];$

$PMIN1 = assocSQL[[1]];$

$Return[PMIN1]$

]

The listSIM-ipo for the 2D-Circle is as the listSIM-ipo of §1.10.1, when you delete zs, ze, zn; listSIM[[nl,4]] becomes listSIM[[nl,3]] and $\{x_n, y_n, z_n\} = \{listSIM[[nl,1]], listSIM[[nl,2]], listSIM[[nl,3]]\}$ becomes $\{x_n, y_n\} = \{listSIM[[nl,1]], listSIM[[nl,2]]\}$.

We apply the listSIM-ipo with listSIM=listSIMBC and listSIM=listSIMVC. In both cases, the

ratio $\frac{npuls}{ninterrupt}$ equals one, in the former case the % gain equals 60 and in the latter case

the % gain equals 92. So, the 2D-VIRT-ipo of a circle with radius equal to 10 000 is still very good, and that was not the case for the 3D VIRT-ipo.

So, Bresenham's classical Circle algorithm, cascaded with the PRM-cs, becomes a perfect constant feedrate algorithm.

The results $\{MaxErr, Cost, npuls, N[\frac{\pi * R}{2}], nkey\}$ for the three circle-ipo's all equal

$\{0.499962, 973.505, 15708, 15708, 14143\}$ and the final ninterrupt of the listSIM-ipo equals 15708.

Note, we can create the array listSIM for every curve, and the results at the sampling points will be perfect, provided we measured the length at the sampling points accurately, but that listSIM is not a discrete curve where the error of every point is smaller than MaxErr.

2. Constant Speed NURBS

In the introduction, we found very fast constant speed IPO's for implicit curves. It is generally accepted that parametric curves can be drawn easier than implicit curves, although everyone prefers the faster circle LSD-ipo of §1.10.2. above the circle NURB-ipo. We can

make a full R-circle NURB with 9 points, a knot vector with 3 double knots and a weight vector such that the weights of the intermediate points equal $w_i = 1/\sqrt{2}$ (Piegle [36] gives a full circle with 6 points). So, with

pts= { {R,0}, {R,R}}, {0,R}, {-R,R}, {-R,0}, {-R,-R}, {0,-R}, {R,-R}, {R,0} },

knots={ 0, 0, 0, 0.25, 0.25, 0.5, 0.5, 0.75, 0.75, 1, 1, 1 },

weights={ 1, w_i , 1, w_i , 1, w_i , 1, w_i , 1 },

fcircle = BSplineFunction[pts, SplineDegree → 2, SplineKnots → knots, SplineWeights → weights],

listxyT = Table[fcircle[t], {t, 0, 1, $\frac{1}{K}$ }] with K the number of subdivisions of the full circle.

The points { $x[i]$, $y[i]$ } of listxyT, for $i \in \{1, K+1\}$ are the sampled data points and the fixed sampling time T_s equals $1/K$. The array listxyT is not a major axis generated list, and cascading the listxyT-ipo with the PRM-cs does not convert this sampled data IPO to a constant speed IPO. The line segments of listxyT can easily be drawn with constant speed using the PRM-cs for each line segment, but the chord error must be smaller than a given maximum error ε_{MAX} and at the sampling points, the tangent is not continuous. Chapter 3.5.3 of [11] describes the sampled-data interpolation method for NURBS, and they conclude that “Due to the demand for high-speed machining, the sampled-data interpolation method is typically used in today’s CNC system.”

We will use the reference pulse method instead of the sampled-data method, hence high-accuracy machining poses no problems, but we have to show that our method is suitable for high-speed machining.

The circle NURB with 9 points is identical with the composite rational Bézier curves

$\left\{ \frac{x_k[\tau]}{w_k[\tau]}, \frac{y_k[\tau]}{w_k[\tau]} \right\}$, and we compute the conic $F_k[x, y]$ using the method of implicitization [37]

with the resultant $F_k[x, y] = \frac{\text{Resultant}[x * w_k[\tau] - x_k[\tau], y * w_k[\tau] - y_k[\tau], \tau]}{(\text{PolynomialGCD}[x_k[\tau], y_k[\tau]])^2} \equiv x^2 + y^2 - R^2$.

We can discretize the conics [6] very fast and with constant speed using the PRM-cs. We calculate the monotonic segments (the extreme points) using the NURB, instead of using the implicit equations, and that is really a big simplification.

Fig. 7 The five conics

Fig. 8 The 3rd-degree and 2nd degree NURB

As an example, we take the trident of [13]. This trident is 3rd-degree NURB fnurb3, and we converted this NURB to a rational 2nd-degree using the method of [31, 32, 33, 34, 35]. So, the 3rd-degree NURB was approximated to a 2nd-degree NURB fnurb2, and we can always do it when we apply algorithm 6 of [35] for a given maximum approximation error ε_{MAX} . The NURB fnurb2 is identical with the five composite rational Bézier curves CB_k[t] (Appendix 4.1, Convert NURB to Conics).

Fig. 7 and Fig. 8 show the results.

The scale factor SC=1000 converts the dimensions of the Trident from mm to μm , Peng [13, Table 1] uses the maximum chord error of 1 μm , and the maximum error MaxErr of the MIDP-ipo is 0.5 μm .

pts3=SC*{ p0, p1, p2, p3, p4, p5, p6 } = SC*{ {20,0}, {40,40}, {24,16}, {20,40}, {16,16}, {0,40}, {20,0} },

knots3={ 0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1 },

weights3={1, 1, 1, 1, 1, 1, 1},

fnurb3= BSplineFunction[pts3, SplineDegree → 3, SplineKnots → knots3, SplineWeights → weights3],

listxyT3 = Table[fnurb3[t], {t, 0, 1, $\frac{1}{K}$ }], LS3 = ArcLength[Line[listxyT3]] $\stackrel{K=10000}{=} 97991.2$.

pts2=SC*{ P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10 },

knots2={0,0,0, $\frac{1}{5},\frac{1}{5},\frac{2}{5},\frac{2}{5},\frac{3}{5},\frac{3}{5},\frac{4}{5},\frac{4}{5},1,1,1$ },

weights2={1, ω_1 ,1, ω_3 ,1, ω_5 ,1, ω_7 ,1, ω_9 ,1},

fnurb2= BSplineFunction[pts2, SplineDegree → 2, SplineKnots → knots2, SplineWeights → weights2],

listxyT2 = Table[fnurb2[t], {t, 0, 1, $\frac{1}{K}$ }], LS2 = ArcLength[Line[listxyT2]] $\stackrel{K=10000}{=} 97481.3$.

Fig. 8 shows the contour and the control points of the 3rd-degree Trident fnurb3[t] in orange and the contour and control points of the 2nd-degree Trident fnurb2[t] in blue. The two contours overlap rather well. The piecewise conics fit exactly the 2nd-degree NURB fnurb2. Fig. 7 shows the contour of the 3rd-degree Trident and the five individual conics of the composite Bézier. The lengths of the conics were not calculated, but LS_{npuls}=97518.

2.1 Experimental results

Table 7 shows the results of the constant speed IPO's ($v_{fr} \equiv \text{feedrate}$).

Method	Type	Max Error <	Exec. Time in seconds	LS2=97481 npuls	LS3=97991 nkey	ninterrupt
MIDP-ipo Alaska	Reference pulse	0.5 μm	0.56	97518	87202	Not used
listSIM Alaska	Reference pulse	0.5 μm	0.07	97518	87202	97518
AM [13]	Sampled- data 1ms	1 μm	1.8 for v_{fr} $\leq 80 \text{ mm/s}$	Not used	Not used	Not used
IAM [13]	Sampled- data 1 ms	1 μm	1.9 for v_{fr} $\leq 80 \text{ mm/s}$	Not used	Not used	Not used
listSIM Mathematica	Reference pulse	0.5 μm	132	97518	87202	97518

Table 7

We calculated, with Mathematica, the maximum error MaxErr using the VIRT-ipo, the region boundary $\mathcal{R} \equiv \text{RegionBoundary}[\text{Polygon}[\text{listxyT3}]]$, and the error function $d_{\text{NURB}}\text{Err} := \text{RegionDistance}[\mathcal{R}, \{x_n, y_n\}]$, and the footpoint of the best point $P_n \equiv \{x_n, y_n\}$ was given by $\text{RegionNearest}[\mathcal{R}, P_n]$, hence the tangent, LM_{cs} , and LN_{cs} could be calculated. The maximum error was less than 0.5 μm and for a resolution of 1 μm (SC equal to 1000) the execution time was about a day. The causes are the error function $d_{\text{NURB}}\text{Err}$ and the slow programming language Mathematica. Therefore, we used the relative fast programming language Alaska Xbase++ 1.9 from Alaska Software, Germany. We could easily import the conics calculated with Mathematica and import or export other data. So, we used Mathematica and converted the 3rd-degree NURB to the 2nd-degree Trident NURB. We converted this NURB to 5 conics with coefficients $\{A_k, B_k, D_k, I_k, J_k, M_k\}$, and exported the conics, the startpoints, endpoints, the extreme points (monotonic points), the direction of the conics (CCW or CW) and the determinant of the conics to our Alaska program. This all took, with a lot of graphics, about 9 seconds. The Alaska program created the Trident-ipo using the very fast Berserkless Midpoint Algorithm [6] based on [5]. This program did not calculate the MaxErr, because it does not need an error function, but it created the array $\text{listSIM} \equiv \{\{x_n, y_n, \text{npuls}\}, \{\dots\}, \dots\}$. This array was also exported to Mathematica; hence we could compare the execution time of the listSIM-ipo using Mathematica (132 seconds) and Alaska (0.07 seconds). This clearly shows the difference of the execution speed of the two programming systems.

The MaxErr of the MIDP-ipo is less than 0.5 μm , and the maximum chord error of [13, Table 1] equals 1 μm , and they used a real-time motion control card based on DSP + FPGA for the AM-method and the IAM method [13, Table 2].

We don't compare the preparation time (Appendix 4.1, item 5). The MIDP-ipo interpolation time is three times faster and the maximum error is twice smaller than the AM IPO. The semi real-time listSIM-ipo interpolation time is twenty four times faster than the AM IPO.

Fig. 9

Fig. 9 shows the real-time speed $\frac{\text{npuls}}{\text{ninterrupt}} \leq 1$ along the Trident curve for F_c equal to 1.

The feedrate fluctuation is virtually nil.

3. Conclusion

Current constant speed IPO's, usually, use Sampled-data IPO's, but by adding an ultra-fast 3-lines algorithm "PRM-cs" to the actual midpoint (Bresenham) algorithms, we converted the very fast midpoint-ipo's to very fast constant speed reference pulse IPO's. This applies to 2D-lines, 3D-lines, 2D-curves, and 2D-NURBS. The PRM-cs measures, in real-time, the length of the discrete curve.

We also showed that the actual 3D-line IPO's are, generally imperfect, but the maximum error is bounded. They can be updated to constant speed 3D-line IPO's and high-speed CNC-machining. The perfect 3D-lines should only be considered in projects where the accuracy is prominent.

All existing pulse-rate or PRM-ipo's use a wrong initialization, which deteriorates the accuracy. The new PRM-cs and updated PRM-ipo use the right initialization.

We proposed, also, a benchmark-ipo, "the listSIM-ipo". This constant speed IPO can be used in real-time for every 2D- and 3D-curve.

The constant speed software DDA-ipo's or BRM-ipo's don't make sense and must be replaced with a PRM-cs cascaded with one or two PRM-ipo's or a MIDP-ipo.

The 3rd-degree Trident NURB shows, as well for normal as for high-speed CNC-machining, that the constant speed reference pulse IPO is much better than the existing sampled-data IPO's. That is evidently also true for lines, circles, and conics.

The PRM-cs can be integrated into the existing CNC-tool machines, and the listSIM-ipo can even be integrated into simple CNC-tool machines.

4. Appendix

4.1 Convert NURB to Conics

Algorithm

Convert 3rd-degree NURB to 2nd-degree NURB,
convert 2nd-degree NURB to composite Bézier curve, and finally
convert composite Bézier curve to tangent conics.

We need the 1st, 2nd and 3rd derivative of the rational NURB fnurb3 and fnurb2 with the subroutines {xnurb3, ynurb3} and {xnurb2, ynurb2} (Appendix 4.2). The letter i gives the 0th, 1st, 2nd, and 3rd derivative of numeric coordinates:

dixd[tj_]:=xnurb[pts, d, knots, weights, i][tj], xd[tj] $\equiv$ d0xd[tj],

diyd[tj_]:=ynurb[pts, d, knots, weights, i][tj], yd[tj] $\equiv$ d0yd[tj],

d is the degree of the NURB, xnurb and ynurb can be found

1. Divide the fnurb3 into a 2nd degree NURB fnurb2 with nP equal to 10 and k = 1, 2, 3, 4, 5

a. Compute the inflection points {P₂, P₄, P₆, P₈} as P_{2k}=Round[{x3[t_{2k}], y3[t_{2k}]}] such that
f[t] = d1x3[t] * d2y3[t] - d2x3[t] * d1y3[t], f'[t] = d1x3[t] * d3y3[t] - d3x3[t] * d1y3[t], {t_a, t_b}
belongs to {0, 0.3, 0.5, 0.7, 1}. The results t_{2k} together with the definitions t₀=0, t₁₀=1
give {t₀, t₂, t₄, t₆, t₈, t₁₀}.

{P₂, P₄, P₆, P₈} $\equiv$ {{30200, 27843}, {21421, 30071}, {18579, 30071}, {9800, 27843}}.

b. Compute the intermediate points {P₁, P₃, P₅, P₇, P₉} as the rounded intersection points of
the tangent lines in the points {P₀=p₀, P₂}, {P₂, P₄}, {P₄, P₆}, {P₆, P₈}, {P₈, P₁₀=p₆}. The
results are {P₁, P₃, P₅, P₇, P₉} $\equiv$

{{36116, 32231}, {24659, 23732}, {20000, 32851}, {15341, 23732}, {3884, 32231}}.

c. By construction the NURB fnurb3 is divided into triangle convex segments

{B₀[k], B₁[k], B₂[k]} == {P_{2k-2}, P_{2k-1}, P_{2k}} for k={1, 2, 3, 4, 5}.

d. Convert the 3rd degree NURB to the 2nd degree NURB with parameters

Pts={P₀, P₁, P₂, P₃, P₄, P₅, P₆, P₇, P₈, P₉, P₁₀} with nP=Length[Pts]=11

weights2={1, ω_1 , 1, ω_3 , 1, ω_5 , 1, ω_7 , 1, ω_9 , 1} // unknown weight vector

knots2={0, 0, 0, $\frac{1}{5}$, $\frac{1}{5}$, $\frac{2}{5}$, $\frac{2}{5}$, $\frac{3}{5}$, $\frac{3}{5}$, $\frac{4}{5}$, $\frac{4}{5}$, 1, 1, 1}

fnurb2[t]={x2[t], y2[t]}={d0x2[t], d0y2[t]} and for j={0, 1, 2, 3}

dix2[tj_]:=xnurbi[Pts, 2, knots2, weights2, i][tj],

diyd[tj_]:=ynurbi[Pts, 2, knots2, weights2, i][tj].

e. Compute weights ω_{2k-1} using the extremes M_{2k-1}

• Compute the extremes M_{2k-1} of fnurb3 parallel to $\overrightarrow{B_0[k]B_2[k]} = \overrightarrow{P_{2k-2}P_{2k}}$

$$f[t] = \text{Det} \begin{bmatrix} d1x3[t] & d1y3[t] \\ x_2[k] - x_0[k] & y_2[k] - y_0[k] \end{bmatrix}, \quad f'[t] = \text{Det} \begin{bmatrix} d2x3[t] & d2y3[t] \\ x_2[k] - x_0[k] & y_2[k] - y_0[k] \end{bmatrix},$$

{t_a, t_b} belongs to {0, 0.2, 0.5, 0.8, 1}. The results t_{2k-1} give {t₁, t₃, t₅, t₇, t₉} and the five
extremes M_{2k-1} = {x3[t_{2k-1}], y3[t_{2k-1}]} . The results are {M1, M3, M5, M7, M9}

= {{32194.7, 25927.5}, {25196.4, 25765.9}, {20000, 32000}, {14803.6, 25765.9},
{7805.3, 25927.5}}

and the shoulder point of fnurb2 in that segment

- Compute the weights ω_{2k-1} corresponding with the intermediate points P_{2k-1}

We round these points to 2 decimal points. We can use the formula of algorithm 4 of [35], but we obtain the same result when we match M_{2k-1} with the shoulder point

$$S_{2k-1} = \frac{\frac{B_0[k] + B_2[k]}{2} + \omega_{2k-1} * B_1[k]}{1 + \omega_{2k-1}} = M_{2k-1} \Rightarrow \omega_{2k-1} = \text{Round}\left[\frac{\text{Norm}[M_{2k-1} - \frac{B_0[k] + B_2[k]}{2}]}{\text{Norm}[B_1[k] - M_{2k-1}]}, 0.01\right]$$

The approximation was good and the results are

$$\{\omega_1, \omega_3, \omega_5, \omega_7, \omega_9\} = \{1.88, 1.55, 2.27, 1.55, 1.88\}$$

- f. Compute the extremes of NURB fnurb2, parallel to the x-axis and y-axis

The BRES-ipo of [6], [5] demands monotonic segments, therefore, the extremes must be known. The determination of the extremes of general parabolas and hyperbolas is complex, but for quadratic Bezier curves it's not complex and for NURBS it's even simpler.

Each conic[k], can have a vertical extreme PVk and a horizontal extreme PHk.

For the vertical extremes,

$$f[t] = \text{Det}\left[\begin{pmatrix} d1x2[t] & d1y2[t] \\ 0 & 1 \end{pmatrix}\right], \quad f'[t] = \text{Det}\left[\begin{pmatrix} d2x2[t] & d2y2[t] \\ 0 & 1 \end{pmatrix}\right],$$

the continue f-segments where $f[t] = 0$ are $\{t_a, t_b\} = \{0, 0.2\}$ and $\{0.8, 1\}$. Therefore, we have two vertical extremes $\text{Round}[x2[tk], y[tk]]$ and the results are

$\{PV1, PV5\} = \{\{32945, 28418\}, \{7055, 28418\}\}$. The vertical extremes PV2, PV3, PV4 don't exist, hence we set them equal to $\{\}$.

For the horizontal extremes,

$$f[t] = \text{Det}\left[\begin{pmatrix} d1x2[t] & d1y2[t] \\ 1 & 0 \end{pmatrix}\right], \quad f'[t] = \text{Det}\left[\begin{pmatrix} d2x2[t] & d2y2[t] \\ 1 & 0 \end{pmatrix}\right],$$

$\{t_a, t_b\}$ belongs to $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

Therefore, we have five horizontal extremes $\text{Round}[x2[tk], y[tk]]$ and the results are $\{PH1, PH2, PH3, PH4, PH5\} =$

$\{\{32479, 29063\}, \{25740, 25704\}, \{20000, 32000\}, \{14260, 25704\}, \{7521, 29063\}\}$

- g. Determine the CCW-direction of conics

The Trident starts at P_0 counterclockwise.

Each triangle contains a rational parametric quadratic Bezier curve, and therefore a conic[k], $k = \{1, 2, 3, 4, 5\}$. The area of the triangle determines the CCW-direction of the conic,

If CCW is true when $b\Delta = 1$ and false when $b\Delta = 0$, then

$$b\Delta[B_0[k], B_1[k], B_2[k]] = \text{If}\left[\text{Det}\left[\frac{1}{2} \begin{pmatrix} \text{First}[B_0[k]] & \text{Last}[B_0[k]] & 1 \\ \text{First}[B_1[k]] & \text{Last}[B_1[k]] & 1 \\ \text{First}[B_2[k]] & \text{Last}[B_2[k]] & 1 \end{pmatrix}\right] \geq 0, 1, 0\right]. \text{ The results are}$$

$bCCW[k] = \{1, 0, 1, 0, 1\}$. Note, when the direction of a triangle $[B_0, B_1, B_2]$ is Counter-Clock-Wise then its area is positive.

2. Convert the special 2nd degree NURB fnurb2 to $\frac{nP-1}{2}$ composite Bezier functions $CB_k[t]$.

When we maintain the same choice for the weights, the piecewise conics fit exactly the NURB fnurb2.

3. Compute the five conic[k]={A_k,B_k,D_k,I_k,J_k,M_k} using the resultant method [37]).

The integer coefficients with A_k, B_k, and D_k divisible by 4 are

conic[1] = {5549416316, 2903901356, -4263503692, -56325551197586,
57937838280272, 33255521653234688}

conic[2] = {1021588076, -381397812, -341007048, -17530221101480,
15759353528220, 118670259509110448}

conic[3] = {637178756, -134171296, 0, -12743575155200, 4497477057716,
104424032165920816}

conic[4] = {1021588076, -381397812, 341007048, -23333301888920,
2119071655420, 350793491006821760}

conic[5] = {5549416316, 2903901356, 4263503692, -165651101427438,
-112602309364144, 4406277530847244800}

4. Prepare the variables which take much calculation time

Examples are the determinant Det[k] of $\begin{pmatrix} A_k & D_k & I_k \\ D_k & B_k & J_k \\ I_k & J_k & M_k \end{pmatrix}$, and the control factor

$\lambda_{M}[k, S_x, S_y] = \frac{A_k + B_k - 2 * S_x * S_y * D_k}{4}$. The signs {S_x, S_y} determine the direction of the

monotonic segments of the conic[k]. The conics for k={1,2,3,4,5} have respectively {3, 2, 2, 2, 3} segments. Let seg equal the segment number, then we must prepare S_x[k, seg_i],

S_y[k, seg_i], $b\lambda_{M}[k, seg] = \text{If}[A_k + B_k - 2 * S_x[k, seg] * S_y[k, seg] * D_k \geq 0, 1, 0]$, and

$b\text{Det}[k] = \text{If}[\text{Det}[k] \geq 0, 1, 0]$. The linear case Det[k]≡0 must be filtered out.

In the experimental results, the BRES-ipo only prepared $b\text{Det}[k] \equiv \text{Det}[k] \geq 0$ and the others were calculated online.

5. The prepared data 1 – 4 are transferred to the real-time MIDP-ipo [6]. We prepared the data with Mathematica and [13] prepared his data with Matlab, and executed on the real-time core.

4.2 Compute the 0th, 1st, 2nd, and 3rd numeric derivatives

The wrong computation of the derivatives of a rational NURB is a persisting bug in Mathematica, and it occurs only when the NURB is rational [38]. We use the modified solution of an unknown user.

We use the subroutine xnurb, ynurb and eventually znurb using a Module.

We do this only for the x-derivatives, and when you replace x with y or z and [[All,1]] with [[All,2]] or [[All,3]], then you calculate y- or z-derivatives.

pts is the vector of the control points;

d is the degree of the NURB;

knots is the knot vector of the NURB;

weights is the weights vector of the NURB and Length[pts]==Length[weights]

```

ptsW = pts weights == pts*weights;
e.g.: {w0*P0, w1*P1, w2*P2} == {P0, P1, P2} {w0, w1, w2};
i is the 0th, 1st, 2nd, or 3rd derivative.
xnurb[pts_, d_, knots_, weights_, i_][u_?NumericQ] :=
Module[{ptsW, x, w, xu, wu, x1u, w1u, x2u, w2u, x3u, w3u},
  ptsW = pts weights;
  x = BSplineFunction[ptsW[[All, 1]], SplineDegree -> d,
    SplineKnots -> knots];
  w = BSplineFunction[weights, SplineDegree -> d,
    SplineKnots -> knots];
  xu = x[u];
  wu = w[u];
  x1u = x'[u];
  w1u = w'[u];
  x2u = x''[u];
  w2u = w''[u];
  x3u = x'''[u];
  w3u = w'''[u];
  Switch[i,
    0, Return[ xu/wu],
    1, Return[(x1u*wu - xu*w1u)/wu^2],
    2, Return[1/wu^3*(2 xu w1u^2 - wu (2 x1u w1u + xu w2u) + wu^2 x2u)],
    3, Return[
      1/wu^4*(-6 xu w1u^3 + 6 wu w1u (x1u w1u + xu w2u) -
        wu^2 (3 x1u w2u + 3 w1u x2u + xu w3u) + wu^3 x3u)]
    ]
  ]

```

5. References

- 1 J.E. Bresenham, "Algorithm for Computer Control of a digital plotter," IBM Systems Journal, Vol. 4, pp. 25-30, 1965.
- 2 J.E. Bresenham, "A Linear Algorithm for Incremental Digital Display of Circular Arcs," Communications of the ACM, Vol. 20, No. 2, pp. 100-106, Feb. 1977.
- 3 G. Spür, G. Stute, M. Weck, "Rechnergeführte Fertigung, Fortschritte der Fertigung auf Werkzeugmaschinen 4, Carl Hander Verlag, München Wien 1977, Book Fortdchritte der Fertigung auf Werkzeugmaschinen 4, ISBN 3-446-12325-3. pp. 44 Ducomat CNC, pp. 15 – 103, Beitrage von G. Duelen.
- 4 V. Huypens, "The Sign Corrected Midpoint Decision Variable selects the candidate point with the minimum Euclidean distance to the conic", Geometric Modeling and Imaging," 3rd International Conference GMAI July 2008, IEEE Service Center Order Nr. P3270, ISBN 978-0-7695-3270-7, pp. 15-20, July 2008.
- 5 V. Huypens, "Relative Squared Distances to a Conic", International Journal of Computer Graphics & Animation (IJCGA), 5, No. 1, pp. 17-37, January 2015.
- 6 V. Huypens, "Flowchart Berserkless Midpoint Algorithm version 4", <https://www.researchgate.net/search.Search.html?type=researcher&query=Valere> Huypens, 2015.
- 7 J. Van Aken, M. Novak, "Curve-Drawing Algorithms for Raster Displays," ACM Transactions on Graphics, Vol. 4, No. 2, April 1985, pp. 147-169.
- 8 J.D. Foley, A. van Dam, Feiner, Hughes, etc. "Computer graphics: Principles and practice,"

- Book, 2nd edition, Addison Wesley, reprinted with corrections, July 1997.
 Book, 3rd edition, 2014.
- 9 Y. Koren, "Computer Control of Manufacturing Systems", McGraw-Hill, USA, 1983.
 - 10 Y. Koren, "Interpolator for a Computer Numerical Control System, IEEE Tr. On Computers, vol. C-25, pp. 32-37, Jan. 1976.
 - 11 Suk-Hwan Suh, S. K. Kang, D.H. Chung, I. Stroud, "Theory and Design of CNC systems", Springer Series in Advanced Manufacturing, 2008, Springer-Verlag, Book DOI 10.1007/978-1/84800-336-1.
 - 12 Eric Andres, "Digital Analytical Geometry: How do I define a digital analytical object", 2015]
 - 13 Junquan Peng, Xinhua Liu, Lei Si, and Jingjing Liu, "A Novel Approach for NURBS Interpolation with Minimal Feed Rate Fluctuation Based on Improved Adams-Moulton Method," Mathematical Problems in Engineering, vol. 2017, Article ID 8718760, 10 pages, 2017. <https://doi.org/10.1155/2017/8718760>.
 - 14 J. Wu, H. Zhou, X. Tang, J. Chen, "A NURBS interpolation algorithm with continuous feedrate", Int. J. Adv. Manuf. Technology, Springer, 2012.
 - 15 J.-P. Reveillès, O. Figueiredo, "New Results about 3D Digital Lines," Proceedings of SPIE, September 1995.
 - 16 O. Figueiredo, J.-P. Reveillès, R. D. Hersch, "Digitization of Bézier Curves and Patches using Discrete Geometry," Proceedings of the 8th Int Conf DGCI99, France, Springer-Verlag, pp 388-398, March 1999.
 - 17 J.-P. Reveilles, "Géométrie discrète, Calcul en nombres entiers et algorithmique", PhD thesis, Université Louis Pasteur, Strasbourg, France, 1991.
 - 18 I. Debled-Rennesson, "Etude et reconnaissance des droits et plans discrets," Strasbourg, France, Décembre 1995.
 - 19 W. M. Newman, R. F. Sproull, "Principles of Interactive Computer Graphics," 2nd Ed., McGraw-Hill Inc., pp. 70-71, 1977, Book .
 - 20 R. L. Graham, D.E. Knuth, O. Patashnik, "Concrete Mathematics," 2nd Edition, Addison-Wesley Publishing Company, 23rd Printing, December 2009, ISBN 0-201-55802-5, Book.
 - 21 John Peatman, "The Design of Digital Systems", Chapter 8 Sec. 8-4 Accumulator rate multiplication pp 350-353, McGraw-Hill, ISBN 10: 0070859574, ISBN 13: 978-0070859579, Book.
 - 22 Per Danielsson, "Incremental Curve Generation", IEEE Tr. On Computers, vol. C-19, pp. 783-793, Sept. 1970.
 - 23 Spiros Papaioannou, "Interpolation Algorithms for Numerical Control", Computers in Industry, vol. 1 , no 1, pp; 27-40, July 1979, North-Holland Publishing Company.
 - 24 Kaufman A. and Shimony E., "3D Scan-Conversion Algorithms for Voxel-Based Graphics," Proc. 1986 Workshop on Interactive 3D Graphics, ACM Press, New York, 1986, pp 45-75
 - 25 X-W Liu, K. Cheng, "Three-dimensional extension of Bresenham's algorithm and its application in straight-line interpolation," Proc Instn Mech Engrs, J. engineering Manufacture, 216 Part B: 45963, November 2001.
 - 26 J.-L. Toutant, "Characterization of the Closest Discrete Approximation of a Line in the 3-Dimensional Space", Lecture Notes in Computer Science, Advances in Visual Computing, Volume 4291/2006, ISSN: 0302-9743 (Print) 1611-3349 (Online), Springer Berlin , 2006.
 - 27 C.E. Kim, "Three-dimensional digital planes," Computer Science Technical Report Series, Univ of Maryland, 1981.
 - 28 C.E. Kim, "Three-dimensional digital planes," IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI-6, pp 639-645, 1984.
 - 29 V. Boyer, J-J Bourdin, "A Faster Algorithm for 3D Discrete Lines," Eurographics 1998, 19th annual Conference of the European Association for Computer Graphics, Lisbon, Portugal, 1998.
 - 30 M. A. Saifee, U.S. Mehta, "Design and Implementation of FPGA based G code compatible CC Lathe Controller," IJECET, IAEME Publication India, 7, 87-100, 2006.
 - 31 X-S Gao, S-C Chou, "Implicitization of Rational Parametric Equations," 1992.
 - 32 X-S Gao, M. Li, "Approximate Implicitization of Planar Parametric Curves using Quadratic Splines," MM Research , 21 52-63, 2002.
 - 33 X-S Gao, M. Li, "Rational Quadratic Approximation to Real Plane Algebraic Curves," Proceedings of the Geometric Modeling and Processing (GMP'04) IEEE, 10, 2004.
 - 34 X-S Gao, M. Li, "Rational Quadratic Approximation to Real Algebraic Curves," Computer Aided Geometric Design, Elsevier, 21, 805:28, 2004.
 - 35 M. Li, X-S Gao, S-C Chou, "Quadratic approximation to plane parametric curves and its application in approximate implicitization," Visual Comput., Springer-Verlag: 12, 2006.
 - 36 L. Piegl, "On NURBS: A Survey," IEEE CGA, Vol. 11, Issue 1, Jan. 1991.
 - 37 T.W. Sederberg, "Computer Aided Geometric Design," September 2016, .BYU Scholars Archive Citation Sederberg, Thomas W., Computer Aided Geometric Design (2012). All Faculty Publications. 1. <https://scholarsarchive.byu.edu/facpub/1>
 - 38 <https://mathematica.stackexchange.com/questions/30790/bsplinefunction-derivatives-wrong-if-using-weights>
 - 39 Spiros G. Papaioannou, "A Non-Orthogonal Interpolation Algorithm for NC Machine Tools," Computers in Industry, vol. 6 ,Issue 2 , April 1985, pp. 103-108, no 1, pp; 27-40, Elsevier B.V.