한국 CAD/CAM 학회 논문집
제 2 권 제 4호 1997년 12월 pp. 253-266

가상 측정을 통한 펜슬곡선 추출

박정환*, 김보현**, 최병규***

Pencil Curve Tracing via Virtual Digitizing

Jung W. Park*, Bo H. Kim** and Byoung K. Choi***

ABSTRACT

Pencil-curve machining, which is a single-pass ball-end milling along a concave edge on adie surface, is widely employed in die-surface machining. The cutter-path used for pencil-curve machining, which is the trajectory of the "ball-center point" of a ball-endmill sliding along a concave-edge region on the die surface, is called pencil-curve. Presented in the paper is a pencil-curve tracing algorithm in which "concave-type" sharp edges are computed from a "virtually digitized" model of the tool-envelope surface. The resulting "initial" pencil-curves are then refined by applying a series of fairing operations. Illustrative examples and methods for enhancing accuracy are also presented. The proposed pencil-curve tracing algorithm has been successfully implemented in a commercial CAM system specialized in die-machining and in the CAD/CAM system CATIA®.

***Key words** : Pencil-curve, Virtual digitizing, Z-map, Pencil-curve machining*

1. 서 론

일반적으로, 금형가공에 있어서 펜슬가공(pencil-curve machining)은 금형곡면(die surface)의 오목 모서리(concave-edge)를 따라 가는 펜슬곡선(pencil-curve)을 공구경로(tool-path)로 삼는 볼엔드밀 가공을 의미하며, 자동차 프레스 금형가공에 적용하고 있다. 펜슬가공의 목적은 크게 1) 영역가공(regional milling)시 오목한 모서리 부위에서의 공구 부하를 줄이거나(relief 가공), 2) 중,정삭 등의 영역가공 후 더 작은 공구로 오목한 모서리 부위에 남아있는 피삭재를 제거(잔삭 가공) 하기 위함이다. 일본 토요타 자동차의 경우, 자동차 내판금형의 릴리프용 펜슬가공을 많이 사용하고 있다[1]. 현재 펜슬가공은 금형가공용 CAM 시스템의 중요한 기능으로 자리잡고 있으며, *Cliks*[2](일본), *Z-Master*[3](한국), *Work-NC*[4](프랑스), *Tebis*[5](독일)등의 CAM 시스템을 그 예로 들 수 있다.

본 논문에서는 효율적이고도 안정적으로 펜슬곡선을 찾아내는 방법을 제시하였고, 이 방법은 현재 상업용 CAM 시스템[3] 및 CATIA 시스템(프랑스 Dassault Systems)의 모듈로 구현되어 있다(CATIA의 경우 본 방식으로 계산한 펜슬곡선을 "seed"곡선으로 하여 수치해석적 방식으로 펜슬곡선을 계산하며, 평활화 및 후처리단계는 거치지 아니한다).

2. 문제정의 및 개략적인 과정

펜슬곡선은 형상곡면(part-surface)의 오목 모서리 부위를 따라 움직이는 공구(볼엔드밀)의 볼 중심점이 만드는 궤적이다(Fig. 1). 이때 형상곡면은 매개변수곡면(*trimmed parametric surface*)들로 구성된다. 본 논문에서 "곡면"은, Bezier 또는 NURB 곡면 등의 매개변수곡면 $\mathbf{r}(u, v)$를 일컫는다. 자동차 프레스 금형의 경우, 형상곡면은 수백~수천 개의 곡면으로 구성된다(내판금형의 경우 통상적으로 3,000개 이상의

*정회원, 영남대학교 기계공학부
**학생회원, 한국과학기술원 산업공학과
***종신회원, 한국과학기술원 산업공학과

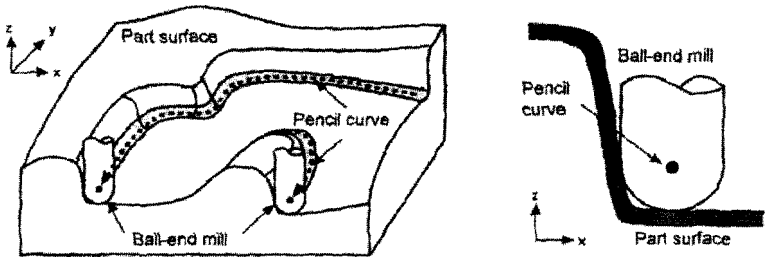

Fig. 1. Pencil-curve tracing problem.

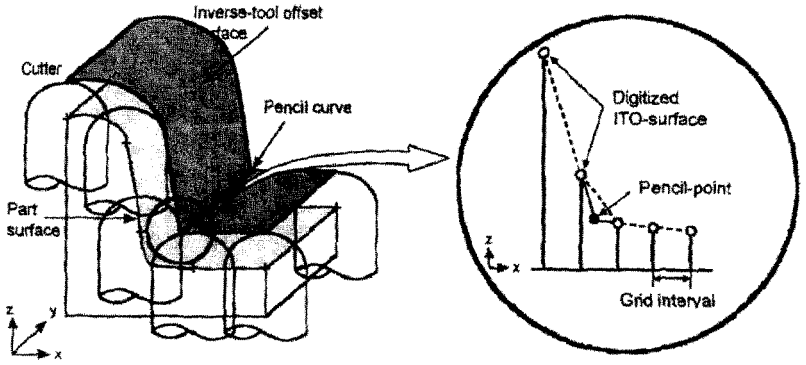

Fig. 2. Concave-edge tracing on the Tool-Envelope Surface (TES).

곡면으로 구성됨). 펜슬곡선 추적 문제는 다음과 같은 상이한 접근 방식이 가능하다.

- SSI(surface/surface intersection) 문제: 옵셋(offset) 곡면간 교선[6] 계산(self-intersection에 의한 교선 포함).
- Circular blending 문제: "rolling-ball" 또는 "ball-position sampling" 방법[7,8]으로 볼 중심점을 수치적으로 계산.
- 각진 모서리(sharp-edge) 탐색 문제: 공구 포락면(tool-envelope surface: TES)의 이산모델(discrete model)로부터 오목한 각진 모서리를 추적.

본 논문은 세 번째 방식을 채택하였는데(Fig. 2), 처음 두가지 방식은 오목 모서리 부위의 필렛반경이 볼엔드밀 공구반경과 비슷할 경우 수치적 특이성(numerical singularity)을 보인다(대부분의 펜슬가공에서 발생함). 세 번째 방식에서 공구 포락면(tool-envelope surface)은 "삼각형 다면체(triangulated facet)" 모델 [9,10] 또는 "z-map" 모델[11]로 표현할 수 있는데, 본 논문은 z-map 모델을 채택 하였다.

펜슬곡선을 추적하기 위한 준비작업으로 우선 공구포락면, 즉 CL(cutter-location) 곡면[11]을 계산한다. 공구포락면은 "공구이동체적(cutter-swept volume)"의 포락(envelope)이며, 형상곡면에 볼 중심점을 놓고 볼엔드밀을 이동(sweeping) 시켰을 때 얻어지는 옵셋 곡면이다. 공구포락면의 수학적 표현이 간단하지 않기 때문에, 본 논문에서는 가상측정(virtual digitizing)에 의해 공구포락면을 계산한 후 이를 z-map으로 저장하도록 하였다. 공구포락면의 z-map(CL z-map) 모델을 얻은 후, 2차원 z-map 배열상에서 오목한 각진 모서리 점(concave sharp-edge point), 즉 펜슬점(pencil-point)을 추적함으로써 초기 펜슬곡선을 구한다.

이후, 초기 펜슬곡선은 공구간섭(cutter interference)[12] 방지 및 평활화(fairing) 과정으로 구성되는 "펜슬곡선 다듬기" 과정을 거쳐 최종적인 펜슬곡선이 된다. 요약하면, 펜슬곡선은 1) 가상측정을 통한 공구포락면 z-map 계산, 2) z-map 상의 오목한 각진 모서리를 추적하여 초기 펜슬곡선 생성, 3) 펜슬곡선 다듬기 과정을 거쳐 얻어진다.

3. 가상 측정에 의한 공구포락면 계산

본 논문에서는 대부분의 프레스 금형에서 그러하듯이, 형상곡면을 위에서 내려다 보았을 때 그림자지는 영역이 없는 경우를 대상으로 한다. 우선, z-

map모델로 표현된 공구포락면을 얻는 방법은 적어도 다음과 같이 3가지가 있을 수 있다. 즉,

- 형상곡면 z-map을 생성한 후, *inverse offset method*[13]를 적용한다.
- 곡면의 옵셋 다면체(offset facet) 모델을 만든 후, 다면체 모델로부터 z-map을 구축한다.
- 형상곡면상에서 볼엔드밀 공구를 이동시키는 모의가공(cutting simulation)[14,15] 방법으로 공 구포락면 z-map을 계산한다.

실제 적용에 있어서 각 방식은 정밀도, 계산시간, 안정성 등의 측면에서 장단점이 다르다. 저자의 구현 결과에 의하면, 첫 번째 방식은 구현이 간단한 반면 상대적으로 정밀도가 떨어지고, 두 번째 방식은 개념적으로 단순한 반면 프로그래밍에 더 많은 노력이 든다. 마지막 방식은 모의가공 알고리즘을 그대로 적용할 수 있으나, 가장 많은 계산시간을 요한다. 본 논문에서는 첫 번째 방식에 의한 공구포락면 z-map 계산만을 설명하도록 한다.

본 논문에서 사용한 "가상측정"이란, "가상" 3차원 측정기(CMM: coordinate measuring machine) 위에 놓여진 곡면(eg, 형상곡면 및 공구포락면) 모델을 "가상" 프로브(probe)로 측정함을 의미한다. 다시 말하면, xy 평면위에서 일정한 간격으로(eg, z-map 격자점) 곡면모델의 z값을 측정(계산) 하고, 이러한 z값의 집합으로 곡면을 표현하는 과정이다(eg, z-map).

3.1 형상곡면의 가상 측정: Z-map sampling

우선 z-map의 2차원 영역(domain)을 정의하기 위하여, x, y 평면위에 형상곡면을 포함하는 "majorizing box"를 정의한다. Majorizing box의 좌하단 모서리 점을(x_o, y_o), 폭은 W, 높이는 H로 놓았을 때 z-map 격자점(grid-point) (i,j)의 좌표(x_i, y_j)는 식(1)로 나타낼수 있다.

$$x_i = x_o + \gamma \cdot i \text{ and } y_j = y_o + \gamma \cdot j$$
$$\text{for } i \in [0, M] \text{ and } j \in [o, N] \quad (1)$$

여기서, γ는 격자간격(grid-interval), i와 j는 2차원 배열의 인덱스(정수), M(=W/γ)과 N(=H/γ)은 z-map 배열의 크기(정수).

다음으로 각각의 곡면 $\mathbf{r}(u, v)=(x(u, v), y(u, v), z(u, v))$에 대해, 분할(trim) 영역을 나타내는 2차원 폐곡선 C를 x, y 평면위에 정의하고, *2D-Jacobian inversion* 알고리즘[11]을 이용하여 폐곡선 C 내부에 위치한 z-map 격자점(x_i, y_j)에서의 곡면 높이값 Z[i, j]를 계산한다. 이 알고리즘이 실패하는 곡면에 대해서는, 곡면을 다면체 모델로 재구성한 후 해당 x, y 위치의 곡면 높이값을 계산한다.

3.2 Z-map 옵셋팅(offsetting)

공구포락면 z-map은 3.1절에서 계산한 형상곡면 z-map Z[i, j]을 옵셋하여 얻을 수 있다. 간단한 z-map 옵셋팅 방식으로는 inverse offset method[11]가 있는데, 각 격자점(m, n)에서 "옵셋 z-map"의 z 값은 식(2)와 같이 나타낼 수 있다[11].

$$Z_o[m,n] = \max\{(Z[i,j]+B(i,j,m,n,R)): (i,j)\in I(m,n,R)\} \quad (2)$$

여기서, $I(m, n, R)=\{(i, j)|((x_i-x_m)^2+(y_j-y_n)^2\}$, $B(i, j, m, n, R)=(R^2-((x_m-x_i)^2+(y_n-y_j)^2))^{1/2}$, x_o, x_m, y_o, y_n은 식 (1)에서, R: 옵셋 거리(또는 볼 반경).

식(2)에서 "$Z[i, j]+B(i, j, m, n, R)$"는 볼 중심점이 p_{ij} = (x_o, y_o, $Z[i, j]$)일 때 격자점(m, n)에서 상반구의 높이(z)값을 나타낸다.

식(2)에 의한 옵셋 z-map 계산은 간단하지만, 그대로 적용하기에는 계산시간이 과도하게 소요된다. 예를 들어, $B(i, j, m, n, R)$의 계산횟수는(W/γ) (H/γ) (2R/γ)^2 이며, 본 논문에서 소개한 실제 예제의 경우 약 130억회가 된다. 따라서, 실제 구현에서는 계산시간 절약을 위한 다양한 방법을 사용하고 있다.

4. 펜슬곡선 탐색 및 추적

본 절은 공구포락면 z-map으로부터 "오목"한 각진 모서리를 탐색·추적하는 과정을 설명한다. 공구포락면 z-map은 볼 옵셋 곡면이므로 "볼록"한 각진 모서리는 존재하지 않는다. Fig. 3에 보인 것은 오목점(concave-point)을 검은 색으로 표시한 수직단면(vertical cross-section: VCS) 이다. 이 수직단면은 i (또는 j)를 "$i=i^*$(또는 $j=j^*$)"로 고정했을 때 YZ평면(또는 XZ평면)상에 정의된 점열 $\{p_{ij}=(x_o, y_j, Z[i,j])\}$로 구성되는 단면으로써, 이후부터 y방향(또는 x방향) 단면이라 부르기로 한다. 우선, 수직단면상의 각 점에 대해 오목각(concave-angle) α를 정의한다(Fig. 3). 이때, 볼록한 점에 대해서는 α = 0로 놓는다.

수직단면상의 연속적인 오목점 중 가장 큰 오목각(α_1)을 가지는 점을 극대오목점(maximal concave-point: MCP)이라 하면, 극대오목점 주변의 4개 오목각(α_j, j=1, 2, 3, 4)은 다음과 같이 정의된다.

- α_1=극대 오목각: 극대오목점의 오목각.

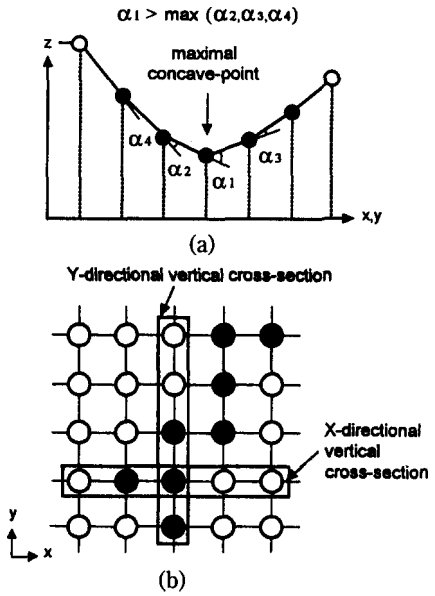

Fig. 3. Construction of a Vertical Cross Section (VCS).

- α_2=이차 극대 오목각(2nd-maximal concave-angle): α_1 옆의 두 오목각 중 더 큰 오목각.
- α_3,α_4 = 이웃 오목각(neighbor concave-angle): α_1, α_2 좌우측에 위치한 오목각.

4.1 펜슬점 탐색을 위한 기본 알고리즘

우선, 수직단면상의 4개 오목각(α_j, j=1, 2, 3, 4)에 근거하여 극대오목점(또는 근처)에 실제 펜슬점이 존재하는 지를 결정하는 과정이 필요하다. 펜슬점을 탐색·추적하는 대상곡면은 "측정"오차가 포함된 이산모델이므로, "펜슬점 탐색문제"는 통계학에서 일컫는 "가설검증 문제(hypothesis testing problem)"와 유사하다고 할 수 있다. 즉, 식(3)으로 표현한 꺾임조건(sharpness condition)이 만족되면 "펜슬점이 존재한다"는 가설을 받아들이는 것이다.

$$(\alpha_1+\alpha_2) > \kappa_s \tag{3}$$

여기서, κ_s는 꺾임기준(sharpness criterion), α_1,α_2는 극대 및 2차 극대오목각.

이때 식(3)의 조건 및 식(4)의 "on-grid" 조건을 동시에 만족하는 경우, 극대오목점은 "on-grid" 펜슬점이 된다(Fig. 4-a).

$$\alpha_1 > \kappa_g \cdot \alpha_2 \tag{4}$$

여기서, "on-grid 기준" κ_g>1.

그렇지 않다면, Fig. 4-b에서 보인 것과 같이 극대오목점과 2차극대오목점 사이에 놓이는 "off-grid" 펜슬점을 다시 정의하고, 이 점에서의 오목각은 α_1+α_2로 놓는다. 본 예에서는 κ_g=4이다.

이제 각각의 펜슬점에는 3가지 속성-진행방향(marching-direction: MD), 측벽위치(wall-location), 품질(quality)-을 부여하며, Backus-Naur 형식으로 표현하면 다음과 같다.

<진행방향> ::="+y"|"-y"|"+x"|"-x"
<측벽위치> ::= "Right"|"Left"|"Undecided"
<품질> ::= "Gold"|"Silver"|"Bronze"|"Clay"

우선, x-방향 수직단면상의 펜슬점은 진행방향이 "+y" 또는 "-y"이며, y-방향 수직단면의 경우는 "+x" 또는 "-x"의 진행방향을 가짐을 알 수 있다. 다음으로, Fig. 4-c의 θ_l, θ_r을 펜슬점에서 만나는 좌측 및 우측 선분에 의한 경사각이라 하자. 진행방향에서 바라 보았을 때 펜슬점의 측벽위치는 다음과 같은 발견적(heuristic) 규칙에 의해 결정할 수 있다. 즉,

"Left" if $\theta_l > \kappa_w \cdot \theta_r$
"Right" if $\theta_r > \kappa_w \cdot \theta_l$
"Undecided" Otherwise (5)

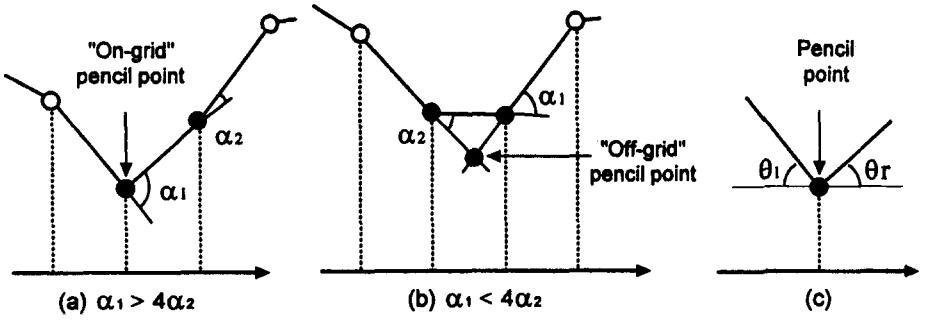

Fig. 4. Determination of 3D pencil-point and wall-direction.

여기서, "측벽위치 기준" $\kappa_w > 1$.

실제 펜슬가공에 있어서 펜슬곡선의 방향은 곡선의 우측에 측벽이 위치하도록 조정함으로써 하향절삭(down-milling)이 되게 하는 것이 바람직하다.

펜슬점의 품질은 "Gold", "Silver", "Bronze", 또는 "Clay"로 등급을 매긴다. 이와같은 분류는 본 논문에서 제시한 방식의 특성에 기인하는 것으로, 옵셋 z-map의 한 단면에서 판단하였을 때 "확실함"의 정도를 의미한다. "Clay"로 판정한 펜슬점들에 대해서는, 5.1절에서 설명하는 "초기 펜슬곡선의 전처리" 단계에서 삭제여부를 최종적으로 판단하도록 하고 있다. 따라서 실질적으로는, "Clay" 이외의 등급은 동일한 등급으로 볼 수 있겠다. 한 펜슬점의 등급을 매기는 과정은 다음과 같은 발견적(heuristic) 규칙을 사용한다.

$$\begin{aligned}&\textit{"Gold" if } \alpha_2=0\\ &\textit{else "Silver" if } (\alpha_3+\alpha_4)/(\alpha_1+\alpha_2) \le \varepsilon_s\\ &\textit{else "Bronze" if } (\alpha_3+\alpha_4)/(\alpha_1+\alpha_2) \le \varepsilon_b\\ &\textit{else "Clay"}\end{aligned} \tag{6}$$

여기서, ε_s는 "silver-point 기준" ε_b는 "bronze-point 기준".

식(3)~(6)에 사용된 각 기준값들은 사용자의 목적에 맞도록 조절이 가능하다. 자동차 프레스 금형 가공의 경우에는 다음과 같은 기준값들을 초기설정값(default value)으로 사용하는 것이 적당한 것으로 판단 된다.

- 꺽임기준(식(3)): $\kappa_c=20^\circ$.
- On-grid 기준(식(4)): $\kappa_g=4$.
- 측벽위치 기준(식(5)): $\kappa_w=2$.
- Silver-point 기준(식(6)): $\varepsilon_s=0.01$.
- Bronze-point 기준(식(6)): $\varepsilon_b=0.07$.

이때 각 기준값의 결정기준은 다음과 같다. 우선 "꺽임기준"의 경우, 일반적인 프레스 금형의 공구포락면을 z-map(격자간격=0.7~1.0 mm)으로 표현하였을 때, 두 오목각의 합이 $\alpha_1+\alpha_2 \ge 20^\circ$를 만족하는 펜슬점만을 의미있는(ie, 가공에 적합한) 펜슬곡선으로 구성하고자 함이다. "On-grid 기준"은 진행상자의 폭을 설정하기 위한 조건이고(4.3절 참조), "측벽위치 기준"은 펜슬곡선의 방향을 조정하기 위한 조건이 되는데, 상기 기준값 자체는 경험적으로 얻은 값이며 펜슬점 추적에 결정적인 영향을 미치지는 않는다.

"Bronze-point 기준(ε_b)" 및 "silver-point 기준(ε_s)" 값은 자동차 프레스 금형의 실 예제(eg, 후드, 도어, 트렁크, 연료탱크등의 내판 금형 모델)로부터 통계적으로 얻은 값이다. 우선 금형모델이 주어지고 공구반경과 필렛반경의 차이를 δ로 두었을 때, $\delta>0$(ie, 펜슬점이 존재하는 경우)인 오목 모서리 $\{E_i: i=1, ..., m\}$가 존재한다. 이때 공구포락면상에 존재하는 각 오목 모서리의 펜슬점들 $\{P_{ij}: j=1, ..., n\}$에 대해 오목각 비율 $\{(\alpha_3+\alpha_4)/(\alpha_1+\alpha_2)\}_j$의 최대값($\Sigma_i$) 및 최소값($\sigma_i$)을 계산할수 있으며, 여러 예제에 대해 Σ_i 및 σ_i의 평균을 계산하여 각각 ε_b(=0.07) 와 ε_s(=0.01)로 설정하였다. 덧붙인다면, "꺽임기준"이 수직단면의 하나 또는 두개의 오목각에 대한 국소적 판단인 반면, 본 기준은 오목각 나열 패턴으로부터 펜슬점 여부를 재검사하는 기준이 된다. 즉, z-map의 특성상 $(\alpha_1+\alpha_2)>\kappa_c$ 인 경우라도 실제 펜슬점이 아닌 경우가 발생하는데, 이를 걸러내기 위한 검사라고 할 수 있겠다.

마지막으로, 식(3)~(6)에서 사용된 오목각 α_i는 z-map 곡면과 수직면(vertical plane)간의 단면위에서 정의된 "근사"각임을 집고 넘어가야 하겠다. 즉, "정확한" 오목각은 점 $\mathbf{p}_{ij}$에서 펜슬곡선에 직교하는 경사면(tilted plane)과 z-map 곡면간의 단면위에서 구하여야 한다.

정확한 오목각을 계산하기 위한 간단한 방법이 Fig. 5에 나타나 있다. Fig. 5-a는 p_{ij}의 근사 오목각 α_i가 x-방향 수직단면상에 정의된 경우이다. 이 그림

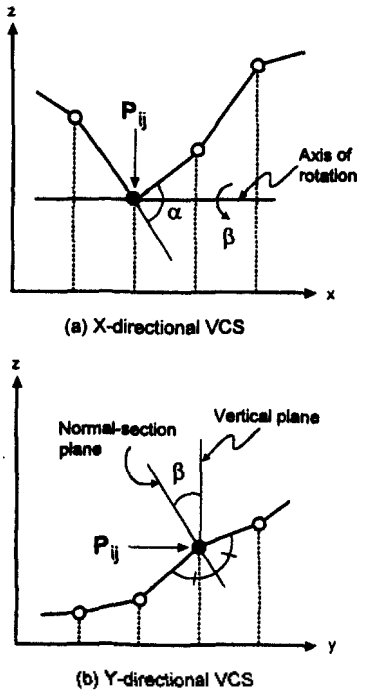

Fig. 5. Concave-angle and tilt-angle.

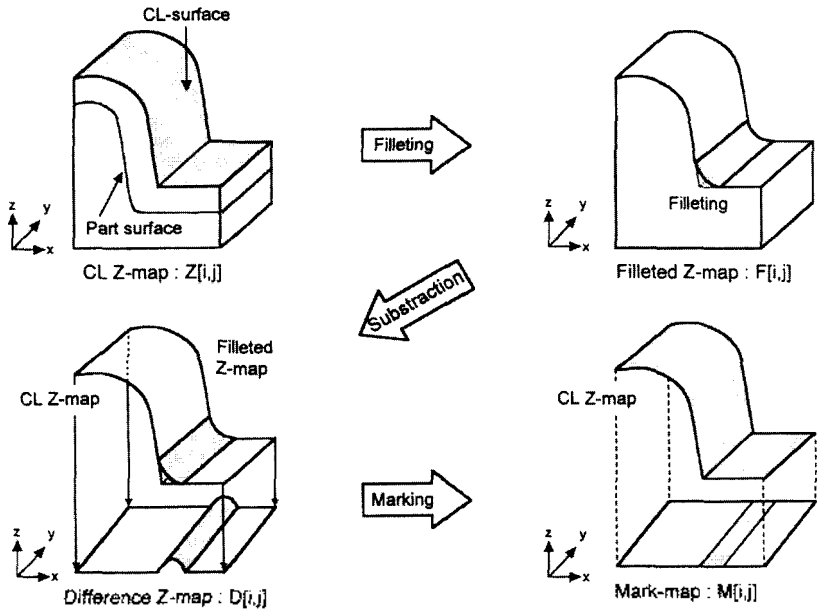

Fig. 6. Steps for generating a mark-map.

에서 보듯이, 회전축(즉, $\mathbf{p}_{ij}$를 지나는 수평선)을 중심으로 수직면을 β_0 만큼 회전이동 시켜 경사면을 정의할 수 있다. Fig. 5-b는 이들 수직면 및 경사면을 yz-평면위의 직선으로 나타내었다. 이 그림으로부터, y-방향 수직단면상의 점 $\mathbf{p}_{ij}$에서 그은 이등분선(bisec- tor line)을 경사면으로 정의하는 것이 적당함을 알 수 있으며, 이때 경사각(tilt angle) β는 이등분선과 수직선간의 각도가 된다. 일단 경사각 β를 알게되면, 근사 오목각 α 대응하는 "정확한" 오목각 α_t는 식(7)에 의해 근사적으로 계산할 수 있다.

$$\alpha_t = 2\tan^{-1}(\tan(\alpha/2)\cdot\cos\beta) \qquad (7)$$

지금부터 식(3)~(6)에서 사용하는 모든 오목각 α_t는 식(7)로 계산한 오목각으로 생각한다.

4.2 초기 펜슬점 탐색

우선 효율적인 펜슬곡선 추적을 위한 준비단계로서, 펜슬점이 존재할 가능성이 있는 모든 오목부위(concave regions)를 2차원 배열 M[*i, j*](mark-map)위에 표시한다. 이러한 mark-map 구축과정은 다음과 같이 정리할 수 있다(Fig. 6).

Construct_mark-map (Z[*i, j*], ρ_{fillet} ⇒ M[*i, j*]);

1. Input: Z-map model Z[*i, j*], Filleting radius ρ_{fillet};
2. Generate a fillet z-map F[*i, j*] ← Upward-offsetting followed by a downward-offsetting using the same offset-distance ρ_{fillet}.
3. Generate a difference z-map D[*i, j*] ← F[*i, j*]-Z[*i, j*] for all *i, j*.
4. For all *i, j* do { if D[*i, j*]> ε and "Z[*i, j*] is concave in x- or y-directional VCS" then mark the grid-point (i.e. set M[*i, j*]=1), else set M[*i, j*]=0 }.

구현된 시스템의 경우 필렛반경값은 $\rho_{fillet}=3\gamma$(γ: 격자간격)이다.

초기 펜슬점을 탐색하는 알고리즘을 설명하기전에, 극대오목점 및 진행방향을 주면 하나의 펜슬점(측벽위치 및 품질 포함)을 찾는 함수를 먼저 보인다. 본 함수는 4.1절의 결과로부터 다음과 같이 정리할 수 있다.

Get_pencil-point(MCP, MD ⇒ Pencil-point, Wall-location, Quality);

1. Input: MCP (maximal concave-point); MD (marching-direction);
2. Compute the "true" concave-angles α_j for *j*=1, 2, 3, 4 at the MCP.
3. If the sharpness condition (3) is not true, then Pencil-point "Null" and stop.
4. If the on-grid condition (4) is true, then Pencil-point MCP, else the Pencil-point is determined from the construction given in Figure 4-b.
5. Determine the Wall-location with respect to the

MD by using the rule (5).

6. Determine the Quality by applying the rule (6).

이제, 초기 펜슬점(initial pencil-point: IPP)을 탐색하기위한 주 알고리즘을 설명하기로 한다. 모든 펜슬점은 "표시 격자점"(marked grid-point, M[i, j]≡1)으로 이루어지는 "표시영역(marked regions)" 내부에 존재 하므로, 펜슬점 탐색은 표시 격자점에서 시작하도록 한다. 즉, 표시 격자점에서 x- 및 y-방향 수직단면상의 극대오목점을 찾는 후, 극대오목점이 존재하는 수직단면을 자세하게 조사하여 초기 펜슬점을 검출한다. 구체적으로, 초기 펜슬점 및 초기진행방향(initial marching-direction: IMD)을 검출하는 과정은 다음과 같다.

Detect_initial-pencil-point(M[*i, j*]⇒IPP, IMD, Wall-location, Quality, M[*i, j*]);

1. Input: mark-map M[*i, j*];
2. Find a marked grid-point from M[*i, j*]; If not found then IPP "Null" and stop;
3. For the marked grid-point, find an MCP from the x-direcitonal VCS and then
   1) set its IMD_x to "+y" or "-y", and
   2) call Get_pencil-point(MCP, IMD_x⇒IPP_x, Wall-locationx, $Quality_x$);
4. For the marked grid-point, find an MCP from the y-direcitonal VCS and then
   1) set its IMD_y to "+x" or "-x", and
   2) call Get_pencil-point(MCP, IMD_y⇒IPP_y, Wall-locationy, Qualityy);
5. If(IPP_x≡IPP_y≡"Null") or ($Quality_x$≡$Quality_y$≡"Clay"), then reset the marked grid-point to zero(M[*i,j*]=0) and go to Step 2.
6. If $Quality_x$ is better than $Quality_y$ then z x, else z y.
7. Return(IPP_z, IMD_z, $Wall\text{-}location_z$, $Quality_z$).

이때, 스텝 3 또는 4에서 극대오목점은 항상 존재할 것이고, 초기진행방향은 임의로 선택할 수 있다(다른 하나의 진행방향으로도 탐색하게 된다). 결과적으로, 상기 알고리즘은 M[*i, j*]상에 표시 격자점이 존재하는 경우 유효한 초기 펜슬점을 검출할 수 있다.

4.3 펜슬곡선 추적

초기 펜슬점에서 출발하여, 더 이상 진행할 수 없을 때 까지 "다음" 펜슬점을 향하여 진행을 계속함으로써 완전한 하나의 펜슬곡선을 얻을수 있다. 다음 펜슬점을 찾기 위하여 "현재" 펜슬점의 앞방향에 "진행상자(marching-cell)"를 정의한다(Fig. 7). 이때

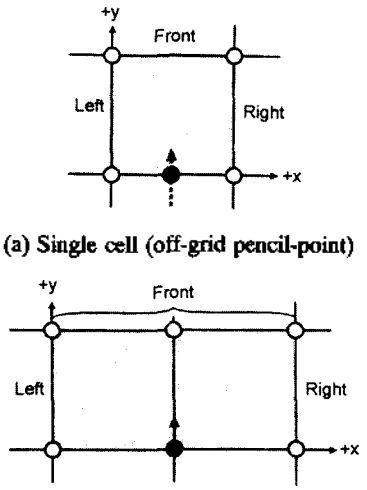

Fig. 7. Marching-cell for pencil-curve tracing (MD="+y").

off-grid 펜슬점에 대해서는 "단일상자(single-cell)"를 정의하고(Fig. 7-a), on-grid 펜슬점은 "이중상자(double-cell)"를 정의한다(Fig. 7-b). 펜슬곡선은 현재 펜슬점을 거쳐 진행상자로 들어간 후, 다음 펜슬점을 지나면서 그 상자(cell)를 벗어난다. 따라서 다음 펜슬점이 존재하는 경우 이 점은 진행상자의 한 변(side)위에 존재할 것이다. 즉, 좌변(Left-side), 상변(Front-side), 또는 우변(Right-side) 중 하나가 된다.

이 3개의 변 중 한 변으로 펜슬곡선이 벗어날 때, "벗어나는 방향"을 각각 Left-Direction(LD), Front-Direciton(FD), Right-Direction(RD)으로 부르기로 하자. 만약 현재 진행방향이 "+y"인 경우(Fig. 7), 벗어나는 방향은 "-x"(LD), "+y"(FD), "+x"(RD)가 된다. 일반적으로, 주어진 현재진행방향(CMD)에 대해 펜슬곡선이 벗어나는 방향은 다음과 같다.

if(CMD≡"+x") then {LD="+y"; FD="+x"; RD="y"}
if(CMD≡"x") then {LD="y"; FD="−x"; RD="+y"}
if(CMD≡"+y") then {LD="x"; FD="+y"; RD="+x"}
if(CMD≡"y") then {LD="+x"; FD="−y"; RD="x"}

일단 다음 펜슬점이 3개 변 중 하나에 존재하면, 이 점을 현재 펜슬점으로 지정하고, 벗어나는 방향을 현재진행방향으로 지정한 후 계속 탐색을 시도한다. 탐색과정에서 찾아낸 펜슬점은 "pencil-curve-list"로 부르는 linked-list 자료구조에 저장하고, 펜슬점을 검출한 수직단면상의 표시 격자점의 표시값은 0으로 만든다.

이제까지의 논의를 바탕으로, 다음 펜슬점으로 한 스텝 진행하는 알고리즘을 적어본다.
즉, 현재 펜슬점 및 현재 진행방향이 주어졌을 때, 다음 펜슬점 및 다음 진행방향을 돌려주는 알고리즘이다. 여기서 사용하는 축약어는 다음과 같다.

- CPP 및 NPP=현재 및 다음 펜슬점(current & next pencil-point).
- CMD 및 NMD=현재 및 다음 진행방향(current & next marching-direction).
- MCP=극대오목점(maximal concave-point).
- LD, FD 및 RD=Left-, Front- 및 Right-direction.
- VCS=수직단면(vertical cross-section).

Marching_next-pencil-point(CPP, CMD⇒NPP, NMD);

1. Input: CPP(current pencil-point); CMD(current marching-direction);
2. For given CPP and CMD, construct a marching-cell and assign its exiting-directions (i.e. the values of LD, FD, and RD) according to(8).
3. If an MCP is found on the Left-side, call **Get_pencil-point**(MCP, LD⇒NPP_l,..).
4. If an MCP is found on the Front-side, call **Get_pencil-point**(MCP, FD⇒NPP_f, ..).
5. If an MCP is found on the Right-side, call **Get_pencil-point**(MCP, RD⇒NPP_r, ..).
6. If no NPP is found from the above three steps, return(NPP="Null") and stop.
7. If more than one valid NPP are obtained, then the one having the same Wall-location as that of the CPP is selected and return its NPP and NMD.
8. Store the CPP in the pencil-curve-list and reset all the marked grid-points located on the current VCS.

상기 알고리즘을 1)다음 펜슬점이 없을때까지 (NPP≡"Null"인 경우), 또는 2)초기 펜슬점을 다시 만날 때 까지(NPP≡IPP인 경우) 되풀이 한다. 이때, 첫 번째 경우에는 초기 펜슬점에서 진행방향을 반대로 놓고 다시 진행을 시작한다. 이제, z-map상의 모든 펜슬곡선을 탐색·추적하는 전체과정은 다음과 같이 정리할 수 있다.

Procedure_pencil-curve-detection-and-tracing( );

1. **Construct_mark-map**(Z[*i*, *j*], ρ⇒M[*i*, *j*]); // ρ= 3γ: fillet radius//
2. **Detect_initial-pencil-point**(M[*i*, *j*]⇒IPP, IMD, Wall-location, Quality, M[*i*, *j*]);
3. *If*(IPP≡"Null") then stop.
4. Pass="First-pass"; //Set flag for restarting the marching operation//
5. CPP=IPP; CMD=IMD; //Set current pencil-point and marching-direction//
6. Repeat {**Marching_next-pencil-point**(CPP, CMD⇒NPP, NMD); CPP=NPP; CMD=NMD;} *until*(NPP≡"Null") or (NPP≡IPP);
7. If(NPP≡IPP) then go to Step 2; //A *closed pencil-curve* has been obtained//
8. If(NPP≡"Null") and(Pass≡"First-pass") then {Pass="Second-pass"; CPP=IPP; CMD=IMD; Go to Step 6} else {Go to Step 2}; //An open pencil-curve//

상기 과정의 결과로서 초기 펜슬곡선 집합을 구할 수 있다. 이 초기 펜슬곡선은 대개 "오류"를 포함하므로, 펜슬가공을 위한 공구경로로 사용하기 위해서는 5절의 "다듬기" 과정이 필요하다.

5. 펜슬곡선 다듬기(refinement)

4절의 "펜슬곡선 탐색·추적" 과정으로 만든 펜슬곡선은 곡선에 존재하는 "떨림현상"(예: 톱날모양의 패턴)으로 인하여 가공에 적합하지 않을 수 있다. 따라서 정밀도를 보장하면서 부드럽고 공구간섭(gouging)이 없는 공구경로를 만들기 위하여 초기 펜슬곡선의 추가적인 다듬기 과정이 필요하다. 이러한 다듬기 과정은[1] 전처리(preprocessing),[2] 점군 평활화 및[3] 후처리(postprocessing) 단계로 이루어진다.

5.1 초기 펜슬곡선의 전처리

전처리는 초기 펜슬곡선 집합으로부터 "유효한" 펜슬곡선을 추출하는 단계이다. 이 과정은[1] "저품질" 펜슬곡선을 없애고,[2] 펜슬곡선의 방향을 조정하고,[3] 곡선 연결 과정으로 세분된다. 우선 펜슬점의 품질이 "Clay"인 경우 그 점을 "clay-point"로 정의하고, 펜슬곡선에서 clay-point가 차지하는 비율을 "clay-ratio"라고 하자. 만약 펜슬곡선의 clay-ratio가 지나치게 높거나 혹은 길이가 짧다면 실 가공에 도움이 되지 아니한다. 따라서 각 펜슬점의 품질 및 측벽위치를 이용하여 전처리에서 다음과 같은 작업을 수행하며, 삭제된 펜슬곡선은 별도의 후처리 없이 완전히 버린다.

1. 연속되는 clay-point의 개수가 n_c 이상인 곡선 단위(segment)는 삭제한다.
2. Clay-ratio가 r_c를 넘는 펜슬곡선은 삭제한다.
3. 길이가 lp 미만인 펜슬곡선은 삭제한다.
4. 곡선 우측에 측벽이 위치하도록(즉, Wall-location≡"Right") 펜슬곡선의 방향을 조절한다.
5. 두 펜슬곡선간의 거리(각 곡선의 끝점간의 3차원 거리)가 l_c 이하면서 동시에 각 끝점의 접선방향(end tangent)이 공선적(collinear)이면, 두 펜슬곡선을 연결한다(concatenation).

이 전처리 단계의 "민감도"(sensitivity)는 "전처리 기준값"의 설정으로 사용자에 의한 조절이 가능하다. 기본 설정값(default value)은 다음과 같으며, 이는 다양한 예제에 대한 테스트 결과를 분석하여 적절하다고 판단한 값이다.

- Clay-segment tolerance: n_c=10.
- Clay-ratio tolerance: r_c=50%.
- Pencil-curve length tolerance: l_p=10γ (γ: 격자간격).
- Concatenation tolerance: lc=5γ.

5.2 펜슬점군 평활화(fairing)

펜슬곡선상의 3차원 점에 대한 평활화를 위하여 펜슬점의 3차원 좌표 $\mathbf{r}_j=(x_j, y_j, z_j)$를 2차원 "영역" 좌표(domain-coordinate) $\mathbf{p}_j=(x_j, y_j)$와 "높이" 좌표(height-coordinate) $\mathbf{q}_j=(s_j, z_j)$로 분리 한다(Fig. 8). 여기서 s_j는 펜슬곡선의 누적길이로서 식(9)에 의해 계산한다.

$$s_j=\sum_{i=1}^{j}|\boldsymbol{p}_i-\boldsymbol{p}_{i-1}| \quad for\ j=1, 2, \cdots \quad with\ s_0=0 \qquad (9)$$

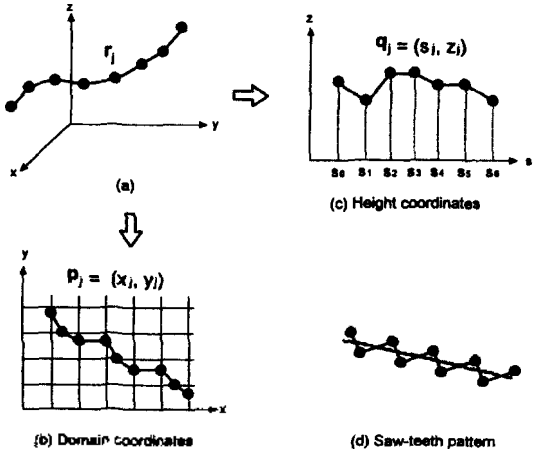

Fig. 8. Decomposition of 3D pencil-points.

본 논문에서 점군 평활화는 차분(difference)을 이용하며, 영역 좌표 $\{\mathbf{p}_j\}$와 높이 좌표 $\{\mathbf{q}_j\}$ 각각에 대해 개별적으로 점군 평활화를 수행한다. 두 경우 모두 평활화는 다음과 같은 두 단계로 이루어 진다.

1. "톱날패턴(saw-teeth pattern)"에 대한 local straightening(Fig. 8-d).
2. 2차원 점군의 global smoothing.

우선, 3차원 점군 $\{\mathbf{r}_j\}$에 대한 n차 차분을 다음과 같이 정의한다.

$$D_j^n=D_{j+1}^{n-1}-D_j^{n-1}\ with\ D_j^0=\boldsymbol{r}_j \qquad (10)$$

n=2일 때 식(10)을 0으로 놓으면, 입력점 $\mathbf{r}_j$의 "이상적인" 위치($\mathbf{r}_j'$)는 다음과 같다.

$$\mathbf{r}_j'=(\mathbf{r}_{j+1}+\mathbf{r}_{j-1})/2. \qquad (11\text{-a})$$

마찬가지로, n=4일 때 식(10)을 0으로 두면 "이상적인" 위치($\mathbf{r}_j'$)은 다음과 같다.

$$\mathbf{r}_j'=(\mathbf{r}_{j+1}+\mathbf{r}_{j-1})/2+[(\mathbf{r}_{j-1}-\mathbf{r}_{j-2})+(\mathbf{r}_{j+1}-\mathbf{r}_{j+2})]/6 \qquad (11\text{-b})$$

상기 2차 차분 평활화(식 11-a) 및 4차 차분 평활화(식 11-b)의 실질적인 의미를 Fig. 9에 보이고 있다. 즉, 2차 차분 평활화는 곡선을 선형화 하고, 반면에 4차 차분 평활화는 간격이 균일한 점군곡선의 곡률(curvature)을 선형화 하는 효과를 가진다고 하겠다. 간혹 식(11-b)의 제곱합(sum-of-squares)과 유사한 양을 점군곡선의 global smoothness measure로 사용하기도 한다[18]. 저자의 경우 local-straightening에

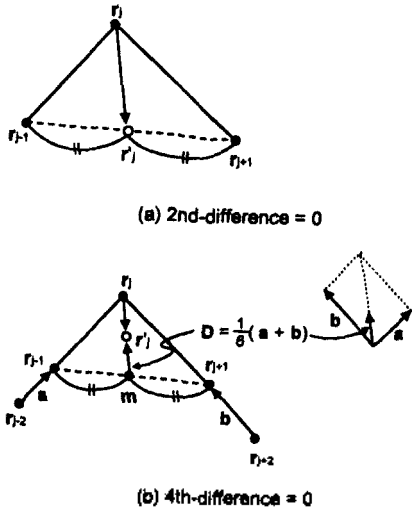

Fig. 9. Physical meaning of the difference fairing.

2차 차분 평활화를 사용하고, global-smoothing에는 4차 차분 평활화를 사용하였다.

하지만 실제 입력접열 (펜슬곡선상의 점열)의 간격이 균일하지 않으므로, 상기 평활화 식을 사용하려면 현길이(chord-lenght)에 대한 정규화(normalization)가 필요하다. 이를 위하여 다음과 같은 현길이를 정의하기로 한다.

$d_{-2}=|\mathbf{r}_{j-2}-\mathbf{r}_{j-1}|;\ d_{-1}=|\mathbf{r}_{j-1}-\mathbf{r}_j|;\ d_{+1}=|\mathbf{r}_{j+1}-\mathbf{r}_j|;\ d_{+2}=|\mathbf{r}_{j+2}-\mathbf{r}_{j+1}|$

그러면, 식(11)은 다음과 같은 정규화된 평활화 식으로 만들 수 있다.

$$r_j' = \left[\frac{d_{-1}}{d_0} r_{j+1} + \frac{d_{+1}}{d_0} r_{j-1}\right]/2 \equiv m \qquad (12\text{-a})$$

$$r_j' = m + \left[\frac{d_0}{d_{-2}} r_{j-1} - r_{j-2} + \frac{d_0}{d_{-2}} r_{j+1} - r_{j+2}\right]/6 \qquad (12\text{-b})$$

여기서, $d_0=(d_{-1}+d_{+1})/2$ 이다.

이때, 식(12)는 $d_{-2}=d_{-1}=d_{+1}=d_{+2}$ 일 때 식(11)과 동일함을 알 수 있다.

이제 식(12)에 의해 $\mathbf{r}_j'$을 계산하면, 실제로 입력점이 이동하는 "조정"위치 $\mathbf{r}_j''$은 이상적인 위치 $\mathbf{r}_j'$과 입력 위치 $\mathbf{r}_j$의 선형조합을 취하여 계산한다. 즉,

$$\mathbf{r}_j''=\mathbf{r}_j+\Phi\cdot(\mathbf{r}_j'-\mathbf{r}_j) \text{ subject to } |\mathbf{r}_j''-\mathbf{r}_j'|\le\tau, \qquad (13)$$

여기서, Φ는 damping factor: $\Phi\in[0, 1]$, τ는 fairing tolerance.

식(13)의 조작은 "감쇄조정(damping correction)"이라고도 하며, Φ는 0.4~0.6의 범위에 두는 것이 적절하다고 판단되었다.

5.2.1 영역좌표 평활화

영역좌표 $\{\mathbf{p}_j\}$에 대한 평활화의 경우, 이상적인 위치 $\mathbf{p}_j'$은 정규화된 평활화식(12)으로 계산한다. 즉,

$$p_j' = \left[\frac{d_{-1}}{d_0} p_{j+1} + \frac{d_{+1}}{d_0} p_{j-1}\right]/2 \equiv m \qquad (14\text{-a})$$

$$p_j' = m + \left[\frac{d_0}{d_{-2}} p_{j-1} - p_{j-2} + \frac{d_0}{d_{+2}} p_{j+1} - p_{j+2}\right]/6 \qquad (14\text{-b})$$

여기서, $d_0=(d_{-1}+d_{+1})/2$ 이다.

한편 $\{d_j\}$는 다음과 같다.

$d_{-2}=|\mathbf{p}_{j-2}-\mathbf{p}_{j-1}|;\ d_{-1}=|\mathbf{p}_{j-1}-\mathbf{p}_j|;$

$d_{+1}=|\mathbf{p}_{j+1}-\mathbf{p}_j|;\ d_{+2}=|\mathbf{p}_{j+2}-\mathbf{p}_{j+1}|$

2차 차분 평활화 식(14-a)는 펜슬점의 영역좌표를 국소적으로 펴는데 사용되며, 4차 차분 평활화 식(14b)는 global-smoothing에 사용된다. (14a)및(14b) 모두, "조종"위치 $\mathbf{p}_j''$를 얻기 위하여 $\mathbf{p}_j'$ 및 입력점 $\mathbf{p}_j$에 대해 식(13)에 의한 감쇄조정을 적용한다.

$$\mathbf{p}_j''=\mathbf{p}_j'+\Phi_d\cdot(\mathbf{p}_j-\mathbf{p}_j') \text{ subject to } |\mathbf{p}_j''-\mathbf{p}_j'|\le\tau \qquad (15)$$

여기서, Φ_d는 damping factor, τ_d는 fairing tolerance.

5.2.2 높이좌표 평활화

높이좌표 평활화의 특징은 1)좌표 $\{\mathbf{q}_j=(s_j,z_j)\}$의 높이값 $\{z_j\}$만을 움직이고, 2)정규화된 평활화 식은 식(9)로 정의된 2차원 영역상의 현길이(domain chord-length) $\{s_j\}$를 사용한다는 것이다. 우선 $\{d_j\}$를 다음과 같이 정의하자.

$d_{-2}=|s_{j-2}-s_{j-1}|;\ d_{-1}=|s_{j-1}-s_j|;$

$d_{+1}=|s_{j+1}-s_j|;\ d_{+2}=|s_{j+2}-S_{j+1}|$

이제 식(12)로부터 다음과 같이 $\{\mathbf{q}_j=(s_j,z_j)\}$에 대한 정규화된 평활화 식을 구할수 있다.

$$z_j' = \left[\frac{d_{-1}}{d_0} z_{j+1} + \frac{d_{+1}}{d_0} z_{j-1}\right]/2 \equiv m \qquad (16\text{-a})$$

$$z_j' = m + \left[\frac{d_0}{d_{-2}}(z_{j-1}-z_{j-2}) + \frac{d_0}{d_{+2}}(z_{j+1}-z_{j+2})\right]/6 \qquad (16\text{-b})$$

여기서, $d_0=(d_{-1}+d_{+1})/2$ 이다.

영역좌표 평활화와 유사하게, "조정" 위치 zj"는 다음과 같은 조정식으로 계산한다.

$$z_j''=z_j'+\Phi_h\cdot(z_j-z_j') \text{ subject to } |z_j''-z_j'|\le\tau_h \qquad (17)$$

여기서, Φ_h는 damping factor, τ_h는 fairing tolerance.

5.2.3 전체적인 평활화 절차

지금까지의 논의에 근거하여, 전체적인 곡선 평활화 과정은 다음과 같이 정리할 수 있다.

1. 식(14-a) 및(15)에 의한 {pj}의 local "saw-teeth pattern" straightening.
2. 식(14-b) 및(15)에 의한 {pj}의 global smoothing.
3. 식(16-a) 및(17)에 의한 {zj}의 local "saw-teeth pattern" straightening.
4. 식(16-a) 및(17)에 의한 {zj}의 global smoothing.

각 단계에서는 조정식(15) 또는 (17)을 반복 적용하게 되는데, 각 반복(iteration)에서는 조정량(deviation)

이 극대(local maximum)인 점만을 움직이고, 더 이상 유의적 개선이 없을 때는 반복을 중단한다.

부연하면, 이상적인 위치 pj'(또는 zj')과 입력점 pj (또는 zj)간의 정규화된 조정량 vj을 다음과 같이 정의하였을 때,

$$v_j=|p_j'-p_j|/|p_{j+1}-p_{j-1}| \text{ 또는 } v_j=|z_j'-z_j|/|z_{j+1}-z_{j-1}| \quad (18)$$

local straightening의 경우, j번째 입력점은 다음 조건이 만족될 때에만 조정된다. 즉,

$$v_j \geq \max(v_{j-1}, v_{j+1})$$

또한 global smoothing의 경우, j번째 입력점은 다음 조건이 만족될 때에만 조정된다. 즉,

$$v_j \geq \max(v_{j-2}, v_{j-1}, v_{j+1}, v_{j+2})$$

5.3 후처리

전처리 및 평활화 과정이 완료된 펜슬곡선에 대해서는 실제 펜슬가공을 위한 추가적인 처리가 필요할 수 있다. 이러한 처리로는 1) 스플라인 곡선 보간(spline-curve fitting), 2) 재분할(remeshing), 3) 공구간섭검사(gouge-checking)등이 있다. 우선, 펜슬곡선을 정의하는 점열 데이터는 3차 스플라인 곡선 내지는 rational B-spline으로 보간할 수 있다[19, 20].

일단 펜슬곡선을 복합 매개변수 곡선(composite parametric curve)로 나타내면, 본 곡선으로부터 균일한 3차원 점들을 다시 추출함으로써 "재분할" 된 점군 펜슬곡선을 얻는다(이때, 점간거리=γ/2(γ:격자간격)). 그리고, 이 점군 곡선은 공구간섭검사 후 일정한 공차내에서(설정값=0.001 mm) 구분적 선형근사(piecewise linear approximation)에 의해 추가적으로 점데이터를 줄이도록 한다.

마지막 단계는 펜슬곡선상의 각 점(펜슬점)에 대한 간섭검사로서, 펜슬점을 볼엔드밀 중심점으로 놓고 공구 바닥면과 형상곡면간의 간섭체크를 수행한다(Fig. 10). 이때 공구간섭이 발견되면, 간섭이 없어지도록 공구를 들어주도록 하고 있다.

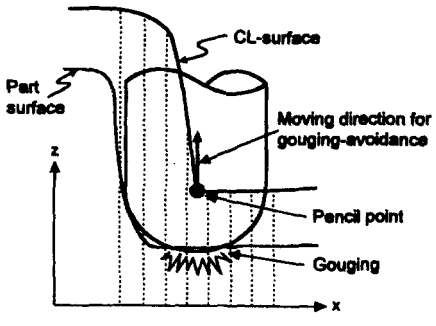

Fig. 10. Gouge checking.

6. 정밀도 향상

본 논문에서 제시한 펜슬곡선 추적 방법은 현재 여러 프레스 금형가공 업체에서 사용되는 상업용 CAM 시스템[3] 및 CATIA 시스템에 구현되어 사용중이다. 한가지 명백한 사실은, 본 방식에 의해 얻어진 펜슬곡선의 품질은 z-map의 격자간격(γ)에 영향을 받는다는 것이다. 즉, γ를 줄이면 정밀도는 높아진다. 그러나 γ가 감소하면, 계산시간 및 메모리 요구가 급격히 증가한다. 이때, 일정한 크기의 형상곡면에 대해, z-map 모델의 메모리 사용은$(1/\gamma)^2$에 비례하고, z-map 옵셋팅 계산시간은 근사적으로$(1/\gamma)^4$에 비례한다. 더구나 단일 z-map의 최대 가용 메모리는, 100MB의 주 메모리를 갖춘 EWS(Engineering Workstation)의 경우 약 40MB가량이다(옵셋팅시 동시에 2개의 z-map 모델에 대한 메모리가 요구된다). 이러한 이유로, 자동차 프레스 금형 가공시 γ는 금형의 크기에 따라 현실적으로 0.4 mm~0.8 mm정도로 설정되어 있다.

그러나, 이와같은 "허용가능" 격자간격(0.4<γ<0.8)은 잔삭가공용 펜슬곡선 생성에 부적합할 수 있다는 문제가 생긴다. 이러한 문제는 다음과 같은 방법으로 해결할 수 있는데, 1) 상기 펜슬곡선을 "seed" 곡선으로 삼아 또다른 수치적 방법(numerical method)를 사용하여 정확한 펜슬곡선을 계산 하거나, 2) 별도의 adaptive sampling scheme을 사용하는 것이다. 현재 CATIA 시스템에서는 첫 번째 방식을 채택하고 있다. 두번째 경우는 적어도 다음과 같은 3가지 방법이 가능한데, 1) quad-tree등을 사용하는 adaptive subdivision scheme, 2) 지정된 영역에 더 작고 정밀한 z-map(일명 core z-map)을 추가하는 방식, 그리고 3) 지정된 영역에 속하는 격자모서리(grid-edge)에서의 추가적인 점데이터 샘플링이 있겠다. 저자는 이 세가지 경우를 모두 고려한 결과 세 번째 방식을 채택하였는 바, 구현이 쉽고(첫번째 경우보다), 계산속도가 빠르다(두번째 경우보다)는 장점을 가진다고 판단하였다.

본 방법의 개념을 x-방향 수직단면에서 보인 결과를 Fig. 11에서 볼수 있다. 우선, 두 격자점을 연결하는 선분을 격자모서리(grid-edge)라고 하자. 수직단면상의 오목점을 연결하는 격자모서리상에서 지정한 개수만큼 2차원 점데이터를 추가적으로 추출하고

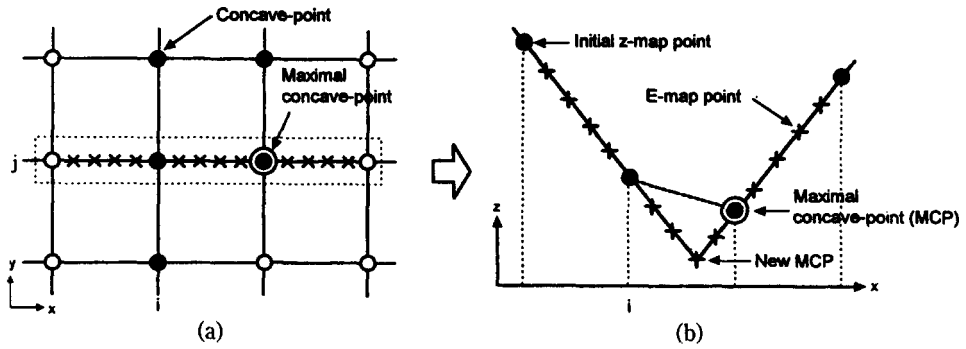

Fig. 11. E-map point sampling for accuracy enhancement.

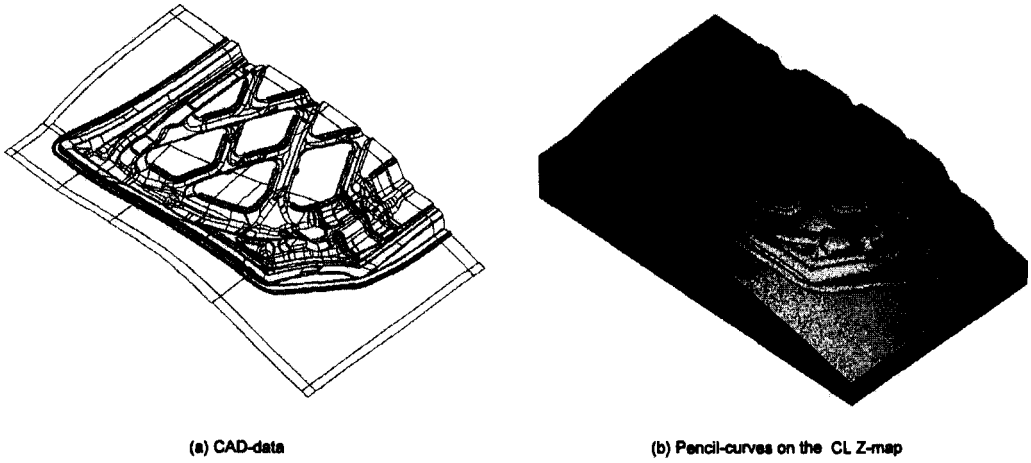

Fig. 12. Pencil-curve tracing for an inner panel stamping-die.

(Fig. 11-a), 각 점에서 공구포락면의 높이값을 측정한다. 이렇게 추가적으로 추출한 높이값을 e-map 점(e-map point)라고 하며, e-map 점을 가지는 z-map은 EZ-map으로 칭한다.

여기서, "펜슬곡선 탐색 및 추적" 알고리즘은 다소간의 수정을 거쳐 Fig. 11-b에 보인 것과 같은 EZ-map의 수직단면에 대해 적용이 가능하며, 생성된 초기 펜슬곡선은 마찬가지로 다듬기 과정을 거치도록 한다. 실제 구현에 있어서는 수직단면상의 각 격자 모서리위에서 9개의 e-map 점을 추가적으로 추출하였고, 이는 대부분의 금형가공시 충분한 것으로 판단 되었다. 결과적으로, 메모리 요구를 다소 늘림으로써(자동차 내판금형의 경우 약 50% 증가) 펜슬곡선의 정밀도는 10배로 증가하게 되는 것이다. 반면 EZ-map을 사용하지 않는 경우, 동일한 정밀도 향상에 요구되는 z-map 메모리는 100배가 될 것이다.

7. 적용 사례

Z-map 펜슬곡선 생성의 한 예를 Fig. 12에 보이고 있다. 본 예제는 승용차 후드 내판금형의 곡면 모델로써, 모두 2,226개의 곡면으로 구성되고, 전체 곡면의 크기는 W=836.5 mm, H=1461.6 mm이다. 격자간격 γ를 0.7 mm로 놓았을 때, z-map 모델은 9.98 MB의 메인 메모리를 필요로 하였다.

본 예제에서는 50Φ 볼엔드밀(볼반경=25 *mm*)에 대한 "릴리프"용 펜슬가공경로를 만들었다. 그 결과, 모두 96개의 펜슬곡선을 얻었으며(Fig. 12-b), IBM RS/6000-365 EWS에서 계산시간은 다음과 같았다.

1) Z-map 옵셋팅: 330 초
2) Mark-map 구축: 186 초
3) 초기 펜슬곡선 생성: 76 초
4) 펜슬곡선 다듬기: 2497 초

즉, 곡면의 z-map 모델에서 시작하여 96개의 곡선을 생성하는데 총 52분 가량이 소요되었으며, 이중 80% 정도는 다듬기 과정에서 소요되었음을 알 수 있다.

8. 결 론

본 논문에서는 금형곡면의 CAD 모델로부터 펜슬곡선을 탐색·추적하는 체계적인 절차를 제시하였다. 본 방식의 특징은 펜슬곡선 추적 문제를 공구포락면에서의 각진 모서리 탐색 문제로 정의하였다는 것이며, 다음과 같은 3개 과정을 거쳐 부드러운 펜슬곡선을 생성하게 된다. 즉, 가상측정에 의한 공구포락면 생성, 초기 펜슬곡선의 탐색 및 추적, 그리고 펜슬곡선 다듬기 과정이다. 제시한 방법의 효용성은 상업용 CAD/CAM 시스템에의 구현을 통하여 검증 하였다. 아울러, 본 논문에서 제시한 오목 모서리 추적 및 점 데이터 평활화 방법은 실제 측정 데이터(예: CMM 데이터)에 대해서도 적용이 가능할 것이다. 마지막으로, 펜슬곡선의 품질 및 계산효율을 향상시키기 위한 추가적인 연구가 필요할것으로 사료된다.

참고문헌

1. Ikemoto, K., *et al.*, "Development & practical application of advanced flexible production system for a year-around continuous operation", *Proc. of IFIP CAPE Conference*, pp. 97-104, 1991.
2. *Cliks User Manual*, ARGO Technos, Japan, 1990.
3. *Z-Master Reference Manual*, Cubic Tek, Korea, 1992.
4. *Work-NC User Guide*, Sescoi, France, 1993.
5. *Tebis-The Complete Solution for Tool, Die, Mold, and Pattern Manufacturing*, Tebis Technishe Informationssysteme, Germany, 1994.
6. Wang, Y., "Intersection of offsets of parametric surfaces", *Computer Aided Geometric Design*, Vol. 13, pp. 453-465, 1996.
7. Klass, R. and Kuhn, B., "Fillet and surface intersections defined by rolling balls" *Computer Aided Geometric Design*, Vol. 9, pp. 185-193, 1992.
8. Choi, B. K., Park, J. W. and Chung, Y. C., "Variable radius blending by ball position sampling", *Proc. of the 1st Pacific Conf. on Computer Graphics & Appl.*, World Scientific Pub Co, pp. 221-234, 1993.
9. Sheng, X. and Hirsch, B. E., "Triangulation of trimmed surfaces in parametric space", *Computer-Aided Design*, Vol. 24, No. 8, pp. 437-444, 1992.
10. Choi, B. K., *et al.*, "Triangulation of scattered data in 3D space", *Computer-Aided Design*, Vol. 20, No. 5, pp. 239-247, 1988.
11. Choi, B. K., *Surface Modeling for CAD/CAM*, Elsevier, 1991.
12. Choi, B. K. and Jung, C. S., "Ball-end cutter interference avoidance in NC machining of sculptured surfaces", *Computer-Aided Design*, Vol. 21, No. 6, pp.371-378, 1989.
13. Takeuchi, Y., *et al.*, "Development of a personal CAD/CAM system for mold manufacturing", *Annals of CIRP*, Vol. 38, No. 1, pp. 429-432, 1989.
14. Jerard, R. B., Drysdale, R. L. and Hauck, K., "Geometric simulation of numerically controlled machining", *Proc. of ASME Int'l Conf. on Computers in Engineering*, ASME, New York, pp. 129-136, 1988.
15. Chung, Y. C. and Choi, B. K., "Non-parametric modeling of cutter swept surfaces for cutting simulation", *Trans of the Society of CAD/CAM Engineers*, Vol. 1, No. 1, Seoul, Korea, pp. 45-55, 1996(In Korean).
16. Faux, I. D. and Pratt, M. J., *Computational Geometry for Design and Manufacture*, Ellis Horwood, 1980.
17. Renz, W., "Interactive smoothing of digitized point data", *Computer-Aided Design*, Vol. 14, No. 5, pp. 267-269, 1982.
18. Eck, M. and Jaspert, R., "Automatic fairing of point sets" in *Designing Fair Curves and Surfaces* (Sapidis ed.), SIAM, pp. 45-60, 1994.
19. Chou, J. J. and Piegl, L. A., "Data reduction using cubic rational B-splines", *IEEE CG & A*, pp. 60-68, May, 1992.
20. Fang, L. and Gossard, D. C., "Multidimensional curve fitting to unorganized data points by nonlinear minimization", *Computer-Aided Design*, Vol. 27, No. 1, pp. 48-58, 1995.

박 정 환

1987년 서울대 물리학과 학사
1990년 KAIST 산업공학과 석사
1995년 KAIST 산업공학과 박사
1995년 ~ 1997년 Chrysler Technology Center 연구원
1997년 ~ 현재 영남대학교 기계공학부 전임강사
관심분야 : CAD/CAM, 5축 NC 가공, Surface Modeling

김 보 현

1991년 전남대 산업공학과 학사
1993년 KAIST 산업공학과 석사
1993년 ~현재 KAIST 산업공학과 박사 과정
관심분야 : CAD/CAM, CAPP, NC Machining

최 병 규

1973년 서울대 산업공학과 학사
1975년 KAIST 산업공학과 석사
1982년 미국 purdue대 산업공학과 박사
1982년 ~현재 KAIST 산업공학과 교수 및 KAIST CIM 연구센터장
관심분야 : Surface Modeling, CAD/CAM, CAPP, 자동화 제조시스템 모델링 및 시뮬레이션