DIFFSD: 미분가능한 부호화 거리 소프트웨어 개발 기술서

김현수
School of Computing
KAIST
Daejeon, South Korea 34141
khskhs@kaist.ac.kr

박진아
School of Computing
KAIST
Daejeon, South Korea 34141
jinahpark@kaist.ac.kr

December 13, 2024

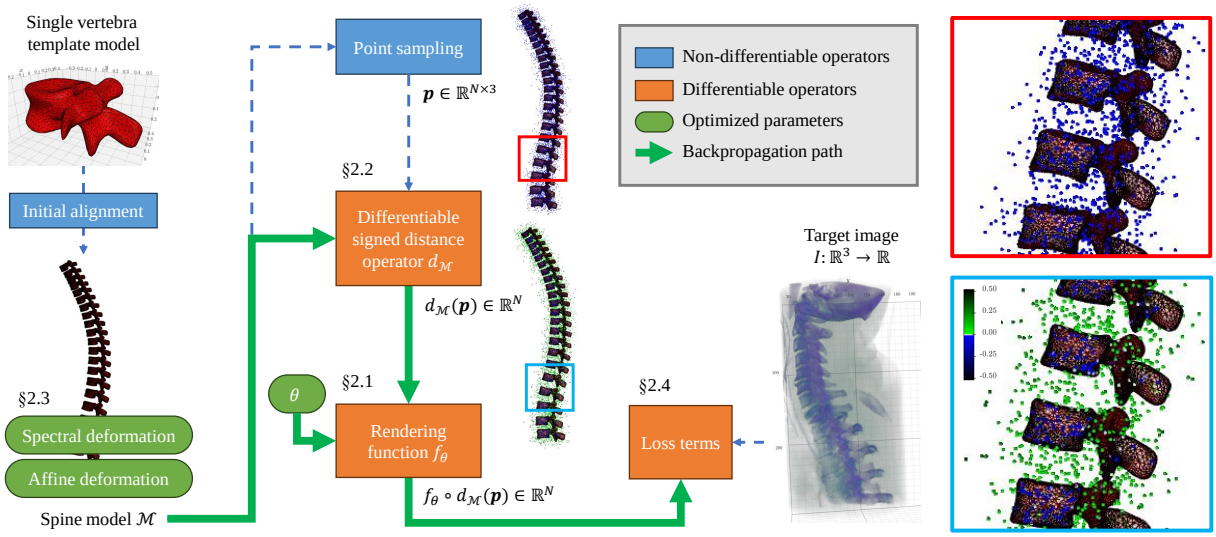


Figure 1: 미분 가능한 부호화 거리(Differentiable signed distance, DiffSD)가 사용된 미분 가능한 외양 모델링(Differentiable appearance modeling) 파이프라인 개요. DiffSD는 척추 모델과 측정 지점들(우측 상단, 빨간 테두리)이 주어졌을 때 부호화 거리(우측 하단, 파란 테두리)를 계산할 수 있고, 부호화거리를 통해 계산된 손실항들로부터 파라미터를 최적화할 수 있도록 역전파할 수 있다.

1 개요

본 기술 문서는 경사 기반 최적화 및 딥러닝 프레임워크에 활용할 수 있는 미분 가능한 부호화 거리(Differentiable signed distance, 이하 DiffSD)의 구현과 활용을 다룬다. DiffSD는 주어진 메시와 측정 지점을 받아서, 각 측정 지점으로부터 메시 표면까지의 거리를 구하고, 측정 지점이 메시의 내부에 있는지에 따라 거리를 음수 또는 양수로 출력하는 부호화 거리(signed distance)를 계산하는 모듈이다. DiffSD는 기존의 부호화 거리 계산 방식에, 주어진 입력에 따른 출력값의 변화량, 즉 경사(gradient)를 analytic하게 계산할 수 있는 역전파 단계가 구현되어 있어 경사 기반 최적화에 사용될 수 있다. DiffSD를 활용한 기법의 예시로, 대량의 의료영상과 레이블링으로부터 패턴을 학습하고 추론하는 딥러닝 방식과 다르게, 구조적 사전지식을 템플릿 모델로 저장하고, 템플릿 모델의 변형과 외양을 주어진 영상에 과적합(overfitting)함으로써 의료영상 분할을 진행하는 DiffAM이 있다 [1].

본 기술문서에서는 미분 가능한 부호화 거리를 계산하는 방법과 이를 최적화하는 방법, 그리고 역전파 단계를 구현하는 방법에 대해 다룬다. 또한, 구현체에 대한 실험 결과와 실제 의료 영상 대상 최적화 문제를 푸는 DiffAM의 결과에 대해 다룬다.

2 방법

2.1 단순 구현

부호화 거리를 계산하는 과정은 기존의 비부호화 거리(unsigned distance)를 구하는 과정과 부호화 거리의 부호를 구하는 과정으로 나뉜다. 자세한 알고리즘은 부록 A의 알고리즘 1에 나와있다. 메시 $\mathcal{M}$와 점 p 사이의 비부호화 거리 $\hat{d}(\mathcal{M}, p)$는 $\min_{q\in\mathcal{M}} |p-q|_2$로 정의된다. 삼각형으로 이루어진 메시의 경우, $\min\{\texttt{TriDist}(f_i, p) | f_i \in \mathcal{M}\}$로 나타낼 수 있으며, $\texttt{TriDist}(f_i, p)$는 삼각형 f_i와 점 p 사이의 거리를 구하는 함수이다.

부호화 거리의 부호를 구하는 과정은, 점 p가 메시 $\mathcal{M}$의 내부에 있는지 판별하여 내부에 있으면 음의 부호를, 외부에 있으면 양의 부호를 부여하는 단계이다. 어떤 점이 메시의 내부에 있는지 판별하는 문제는 Angie weighted pseudonormal [2]과 같은 방법이 있지만, DiffSD는 과도하게 변형된 메시나 불완전한 메시에서도 강건한 3D winding number [3]을 사용한다. Winding number는 직관적으로 어느 점을 기준으로 메시가 감싸고 있는 입체각(solid angle)의 비율로 표현할 수 있다. 예를 들어, 단순한 닫힌 메시는 공간을 winding number가 0인 바깥과 1인 안쪽으로 이분하게 된다. 따라서, 부호화 거리의 부호는 winding number 0.5를 기준으로 부호 함수 sgn 을 사용해 계산할 수 있다. 이를 종합하면, 부호화 거리를 계산하는 과정을 식 1으로 정리할 수 있다.

$$d(\mathcal{M}, p) = \text{sgn}(0.5 - \texttt{WN}(\mathcal{M}, p)) \times \hat{d}(\mathcal{M}, p). \tag{1}$$

메시 $\mathcal{M}$의 삼각형 수가 F이고 부호화 거리를 계산하는 점의 수가 P일 때 알고리즘 1의 순차적 시간복잡도(sequential time complexity)는 $O(PF)$로, 공간복잡도는 $O(P)$로 계산된다. 각 점마다 연산을 병렬화하고 각 삼각형마다 수행되는 $\min, +$을 Parallel reduction으로 병렬화하는 경우의 병렬적 시간복잡도(parallel time complexity)는 $O(\log F)$, 공간복잡도는 $O(PF)$로 나온다. 두 경우 모두, 점과 삼각형이 늘어날수록 시간 또는 공간복잡도가 이차적으로(quadratically) 늘어나기 때문에, 복잡한 문제에서는 활용하기 어려운 문제가 있다. 다음 절에서는 시간 및 공간복잡도를 줄이기 위해 공간자료구조인 Bounding volume hierarchy (BVH)를 활용하는 방법을 다룬다.

2.2 BVH를 사용한 빠른 구현

부호화 거리를 계산하는 과정에 있는 비부호화 거리 계산 과정과 winding number를 활용한 부호 판별 과정 모두 BVH를 통해 최적화가 가능하다. BVH를 활용한 부호화 거리 계산 과정을 보여주는 자세한 알고리즘은 부록 A의 알고리즘 2에 나와있다. 비부호화 거리 계산 과정은 $\min$ reduction 과정이기 때문에 비교적 직관적으로 최적화할 수 있는데, 현재 순회하고 있는 bounding volume까지 거리의 lower bound가 지금까지 찾은 최소 거리보다 큰 경우 해당 bounding volume 및 그 하위 볼륨에 대해 순회를 건너뛰는 방식으로 불필요한 연산을 줄일 수 있다. 이 때, 전역 최소 거리를 빨리 찾을수록 불필요한 순회를 최대한 많이 건너뛸 수 있는데, 이를 위해 하위 볼륨을 순회할 때 점에서 가까운 하위 볼륨을 먼저 순회하는 전략을 취하여 실행 시간을 더 줄일 수 있다. 깊이 우선 순회(Depth first traversal)과 가까운 하위 볼륨 우선 순회를 사용하여 비부호화 거리 계산 과정을 시간복잡도 $O(P \log F)$ 및 공간복잡도 $O(P + F)$까지 최적화할 수 있다.

부호 판별 과정은 주어진 점을 기준으로 모든 삼각형의 winding number를 합하는 과정이 있어 이를 정확히 계산하는 과정은 반드시 모든 삼각형을 순회하는 과정이 필요하다. 하지만, 부호화 거리를 계산하기 위해 winding number를 사용하는 의의는 그 정확한 값을 계산하는 것이 아닌 부호 판별 과정에 있기 때문에, 정밀한 값을 필요로 하지 않는 경우가 일반적이다. 따라서, 점과 3차원 메시의 winding number를 근사할 수 있는 Fast winding number [4]를 사용하여 부호 판별을 sublinear 시간 내에 계산할 수 있다. Fast winding number는 어느 점에 대한 삼각형의 winding number가 점근적으로(asymptotically) 점과 삼각형의 무게중심 사이의 거리의 제곱에 반비례하다는 사실을 이용하여, 점에서부터 멀리 떨어진 삼각형들을 하나의 쌍극자(dipole)로 근사하여 모든 삼각형을 탐색하지 않고도 winding number의 근사치를 구하는 방법이다. 이러한 방식은 일반적으로 Barnes-Hut approximation [5]으로 알려져 있다.

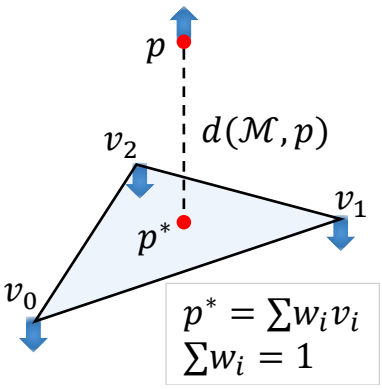
Winding number의 Barnes-Hut approximation 계산을 위한 BVH 구축 과정은 상향식(bottom-up)으로 이루어진다. BVH의 말단 볼륨에 있는 개별 삼각형 $f = (v_0, v_1, v_2)$은 위치 $q = \frac{v_0+v_1+v_2}{3}$이고 벡터 $\bar{n} = \frac{1}{2}(v_2 - v_0) \times (v_1 - v_0)$인 쌍극자로 근사될 수 있으며, 멀리 떨어진 점 p에 대한 winding number는 $\frac{\bar{n}\cdot(p-q)}{4\pi|p-q|^3}$으로 근사된다. 또한, 두 쌍극자 $(q_1, \bar{n_1}), (q_2, \bar{n_2})$가 있을 때, 이들은 하나의 쌍극자 $(\frac{|\bar{n_1}|q_1+|\bar{n_2}|q_2}{|\bar{n_1}|+|\bar{n_2}|}, \bar{n_1} + \bar{n_2})$로 근사될 수 있다. winding number를 근사하는 쌍극자들은 winding number를 구하려는 점의 위치와 관계없이 쌍극자들이 근사하는 삼각형 고유의 특성이기 때문에, 이를 미리 계산할 수 있다. 이를 활용하여 winding number의 근사를 위한 BVH 구축에는 추가 시간복잡도 $O(F)$ 및 추가 공간복잡도 $O(F)$가 필요하며, P개의 지점마다 BVH를 순회하며 winding number를 근사하는 과정은 시간복잡도 $O(P \log F)$ 및 공간복잡도 $O(P + F)$로 최적화된다. 따라서, 비부호화 거리 계산 과정과 부호 판별 과정 모두 BVH를 활용하면 메시의 복잡도에 대해 sublinear하게 연산할 수 있게 된다.

BVH를 활용한 알고리즘을 GPU에서 효율적으로 활용하기 위해서는 BVH의 구축과 순회를 병렬적으로 수행하는 방법이 필요하다. DiffSD는 이를 위해 Morton code를 이용한 병렬 BVH 구축 및 순회 방법 [6]를 사용한다. x좌표가

$0.X_0X_1X_2...$이고 y와 z좌표도 비슷하게 정의된 어느 점에 대해서, Morton code는 $X_0Y_0Z_0X_1Y_1Z_1...$로 정의된다. 서로 다른 위치의 여러 점에 대해 Morton code를 구해 이를 정렬하면, 그 순서는 이진 기수 트리(binary radix tree)의 순회 순서와 같은 순서가 되는데, Karras [6]는 정렬된 Morton code에서 이진 기수 트리 구조를 상향식 분석을 통해 병렬적으로 구하는 알고리즘에 대해 다룬다. DiffSD는 이에 추가로 각 볼륨마다 winding number의 Barnes-Hut approximation을 구하는 과정을 추가로 진행하며, 추가적인 최적화를 위해 각 점마다 반복되지만 점의 위치와 무관한 삼각형 관련 연산들을 미리 계산하는 과정 또한 진행한다. GPU를 활용해 각 점마다의 연산을 병렬적으로 진행하면 병렬적 시간복잡도는 $O(\log F)$, 공간복잡도는 $O(P+F)$로, sublinear 시간 내에 연산이 가능해 연산을 반복적으로 수행하는 경사 기반 최적화 등에 사용하기 적합하다.

2.3 역전파

주어진 메시 $\mathcal{M} = (V, F)$와 점 p 사이의 부호화 거리 d는 식 1과 같이 계산되는데, 이 연산을 경사 기반 최적화에 활용하기 위해서는 해당 연산의 경사를 목표 파라미터에 대한 손실 함수 (loss function) 경사에 반영하는 역전파(backpropagation) 과정이 구현되어야 한다.



부호화 거리의 경사는 기하학적으로 설명될 수 있다. 우측 그림은 부호화 거리의 경사 계산에 의존적인 변수, 점의 위치 p, 점에서 가장 가까운 삼각형 (v_0, v_1, v_2), 그리고 p에서 가장 가까운 메시 $\mathcal{M}$ 위의 점 p^*만을 남긴 그림이다. $\frac{\partial d}{\partial p}$는 p의 변화에 따른 d의 변화 비율을 의미하고, 직관적으로 p가 $p-p^*$ 방향으로 멀어질 때(그림 상의 파란색 화살표 방향) 그 크기만큼 d 또한 증가함을 알 수 있다. 반면, 점 p가 메시 $\mathcal{M}$ 내부에 있을 때에는 반대로 가까워질 때 d가 증가한다.

마찬가지로, 삼각형의 한 정점 v_i의 변화에 따른 d의 변화를 보면, v_i는 $\frac{\partial d}{\partial p}$와 반대 방향으로 움직일 때 d가 증가함을 알 수 있다. 변화량의 비율은 p^*의 barycentric coordinate, 즉 p^*를 v_i의 선형 합으로 나타냈을 때의 계수에 비례함을 간단히 보일 수 있다. 이를 식으로 나타내면 아래와 같다.

$$\frac{\partial d}{\partial p} = \text{sgn}(d)\frac{p-p^*}{|p-p^*|} \qquad \frac{\partial d}{\partial v_i} = -w_i\frac{\partial d}{\partial p}. \tag{2}$$

부호화 거리는 점 p와 메시 $\mathcal{M}$을 같은 크기로 평행이동하여도 그 값이 변하지 않는데, 식 2에서도 $\frac{\partial d}{\partial p} + \sum \frac{\partial d}{\partial v_i} = 0$임을 통해 이러한 특성을 만족하는 경사식임을 보일 수 있다.

DiffSD는 GPU를 활용해 각 점마다 식 2의 계산을 병렬적으로 진행함으로써 경사 기반 최적화에 필요한 역전파를 수행한다. 역전파를 위해서는 모든 점에서 식 2와 같이 계산된 경사를 삼각형 정점 v에 합하는 단계가 필요한데, 이는 GPU의 원자성 연산(atomic operation)을 활용해 병렬적 계산에 오류가 없도록 한다.

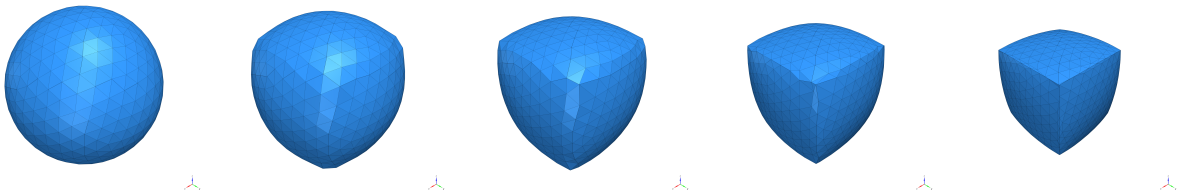
3 구현 및 실험 결과

DiffSD는 Pytorch 프레임워크 [7]를 통해서 활용될 수 있도록 구현되었다. GPU에서 실행되는 코드는 Python 코드를 CUDA 커널로 JIT 컴파일하는 Numba 프레임워크 [8]를 이용하여 구현되었다.

그림 2는 부호화 거리를 거친 역전파를 통한 메시 최적화의 예시이다. 사용된 메시는 삼각형 1280개의 반지름 1인 구형 메시이고, 목표로 하는 메시는 각 변의 길이가 1.5인 육면체이다. 매 반복마다 10000개의 포인트를 샘플링한 뒤, 육면체의 analytic 부호화 거리와 최적화하려는 구형 메시와의 부호화 거리 간의 mean square error를 역전파하여 메시 정점 위치를 최적화하였다. 최적화 방법은 step size 0.1의 stochastic gradient descent를 사용하였다.

그림 3는 BVH를 활용한 빠른 구현의 효율성을 검증하기 위해 단순 구현과의 메모리 사용량을 비교한 그림이다. 공간 복잡도 $O(PF)$의 병렬화 단순 구현체는 초선형적인 메모리 요구로 인해 24GB의 가용 GPU 메모리를 초과하는 경우가 발생하지만, 공간 복잡도 $O(P+F)$의 빠른 구현체는 이러한 문제 없이 모든 테스트 케이스에 대해 메모리 효율적으로 실행이 가능함을 볼 수 있다.

미분 가능한 부호화 거리의 실용적인 활용 예로, 메시의 외양을 렌더링하는 과정에 부호화 거리를 이용하여 예측되는 외양과 실제 측정된 이미지의 손실을 최소화하는 방식으로 메시를 최적화하는 DiffAM [1]이 있다. DiffAM 척추 모델의 3차원 이미지를 예측하기 위해, 이미지를 구성하는 복셀의 위치로부터 척추 모델까지의 부호화 거리를 예측하여 이를 렌더링 함수의 입력으로 사용함으로써 이미지를 예측하는데, 이 과정에서 미분 가능한 부호화 거리를 사용하여서 예측 이미지로부터 계산된 손실을 메시까지 역전파할 수 있다.



(a) (b) (c) (d) (e)

Figure 2: 부호화 거리의 mean square error 손실 역전파를 통해 원본인 구형 메시(2a) 를 주어진 부호화 거리를 가지는 육면체(2e)로 최적화하는 과정.

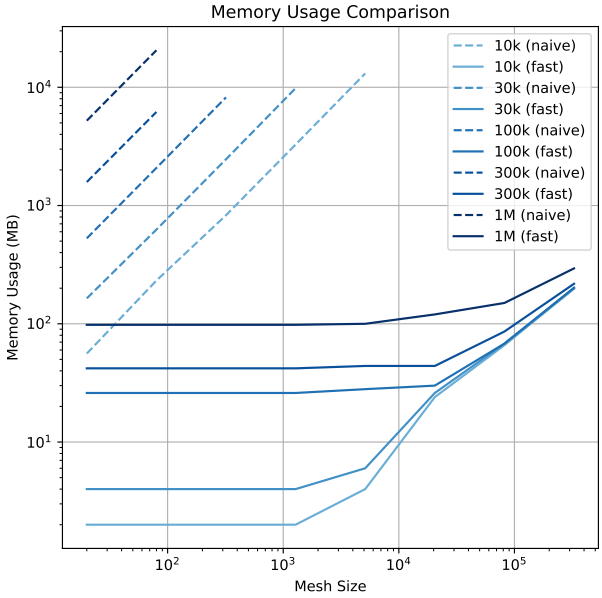


Figure 3: 병렬화한 알고리즘 1과 알고리즘 2의 메시 삼각형 수(가로축) 및 쿼리 포인트 수(범례)에 따른 GPU 메모리 사용량(세로축) 비교.

References

[1] Hyunsoo Kim and Jinah Park. Vertebral segmentation without training using differentiable appearance modeling of a deformable spine template. In Olivier Colliot and Jhimli Mitra, editors, *Medical Imaging 2024: Image Processing*, volume 12926, page 129262N. International Society for Optics and Photonics, SPIE, 2024.

[2] Jakob Andreas Bærentzen and Henrik Aanaes. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):243–253, 2005.

[3] Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

[4] Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[5] Josh Barnes and Piet Hut. A hierarchical o (n log n) force-calculation algorithm. *nature*, 324(6096):446–449, 1986.

[6] Tero Karras. Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*, pages 33–37, 2012.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[8] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

A 알고리즘 상세

Algorithm 1: Naive implementation of signed distance

Input: A mesh $\mathcal{M} = (V, F)$, a list of points $P \in \mathbb{R}^{N\times 3}$

Output: Signed distances $D \in \mathbb{R}^N$

begin

  `/* Compute unsigned distances */`

  for *each point* $p_i \in P$ **do**

    $D_i \leftarrow \infty$;

    for *each face* $f_j \in \mathcal{M}$ **do**

      $D_i \leftarrow \min(D_i, \texttt{TriDist}(f_j, p_i))$;

    end

  end

  `/* Determine the sign */`

  for *each point* $p_i \in P$ **do**

    $w_i \leftarrow 0$;

    for *each face* $f_j \in \mathcal{M}$ **do**

      $w_i \leftarrow w_i + \texttt{WindingNumber}(f_j, p_i))$;

    end

    $D_i \leftarrow D_i \times \text{sgn}(0.5 - w_i)$;

  end

  Return D;

end

Algorithm 2: Faster implementation of signed distance using bounding volume hierarchies

Input: A mesh $\mathcal{M} = (V, F)$, a list of points $P \in \mathbb{R}^{N\times 3}$

Output: Signed distances $D \in \mathbb{R}^N$

begin

  `/* Compute unsigned distances */`

  $H \leftarrow \texttt{BuildBVH}(\mathcal{M})$;

  for *each point* $p_i \in P$ **do**

    $D_i \leftarrow \infty$;

    while *traversing* $h_j \in H$ **do**

      $D_m, D_M = \texttt{BVHDistEstim}(h_j, p_i)$;

      if $D_m \geq D_i$ **then**

        Skip h_j;

      else if h_j *is a leaf node* **then**

        for *each face* $f_k \in h_j$ **do**

          $D_i \leftarrow \min(D_i, \texttt{TriDist}(f_k, p_i))$;

        end

      else

        Traverse the children of h_j;

    end

  end

  `/* Determine the sign */`

  $H_w \leftarrow \texttt{BarnesHutApprox}(H)$;

  for *each point* $p_i \in P$ **do**

    $w_i \leftarrow 0$;

    while *traversing* $h_j \in H$ **do**

      if $\texttt{BVHCenterDist}(h_j, p_i) \geq \texttt{BVHDiameter}(h_j)$ **then**

        $w_i \leftarrow w_i + \texttt{WNApprox}(h_j, p_i)$

      else if h_j *is a leaf node* **then**

        for *each face* $f_k \in h_j$ **do**

          $w_i \leftarrow w_i + \texttt{WindingNumber}(f_k, p_i)$;

        end

      else

        Traverse the children of h_j;

    end

    $D_i \leftarrow D_i \times \text{sgn}(0.5 - w_i)$;

  end

  Return D;

end