Wygładzanie siatek modeli 3D

Damian Wojtyczko

1 Wstęp

Wygładzanie siatek 3D to istotny krok w przetwarzaniu modeli przestrzennych, szczególnie cenny w przypadku modeli pozyskanych przy użyciu skanera 3D. U podstaw algorytmów wygładzania leży matematyczna teoria operatorów różniczkowych, a w szczególności - operator Laplace'a. W tym artykule dokładniej opiszę algorytm wygładzania siatek modeli 3D i omówię jego podstawy matematyczne.

2 Siatki wielokątowe

Siatka wielokątowa to struktura złożona z wielokątów, które są ze sobą połączone krawędziami. Aproksymuje ona powierzchnię żądanego obiektu korzystając z zadanych wierzchołków (punktów), na których tworzona jest siatka (proces ten nazywany jest teselacją). Siatki wielokątowe w informatyce wykorzystywane są m.in. do odwzorowania obiektów 3D w postaci cyfrowej.

Siatka wielokątowa (w grafice 3D) składa się z poniższych elementów:

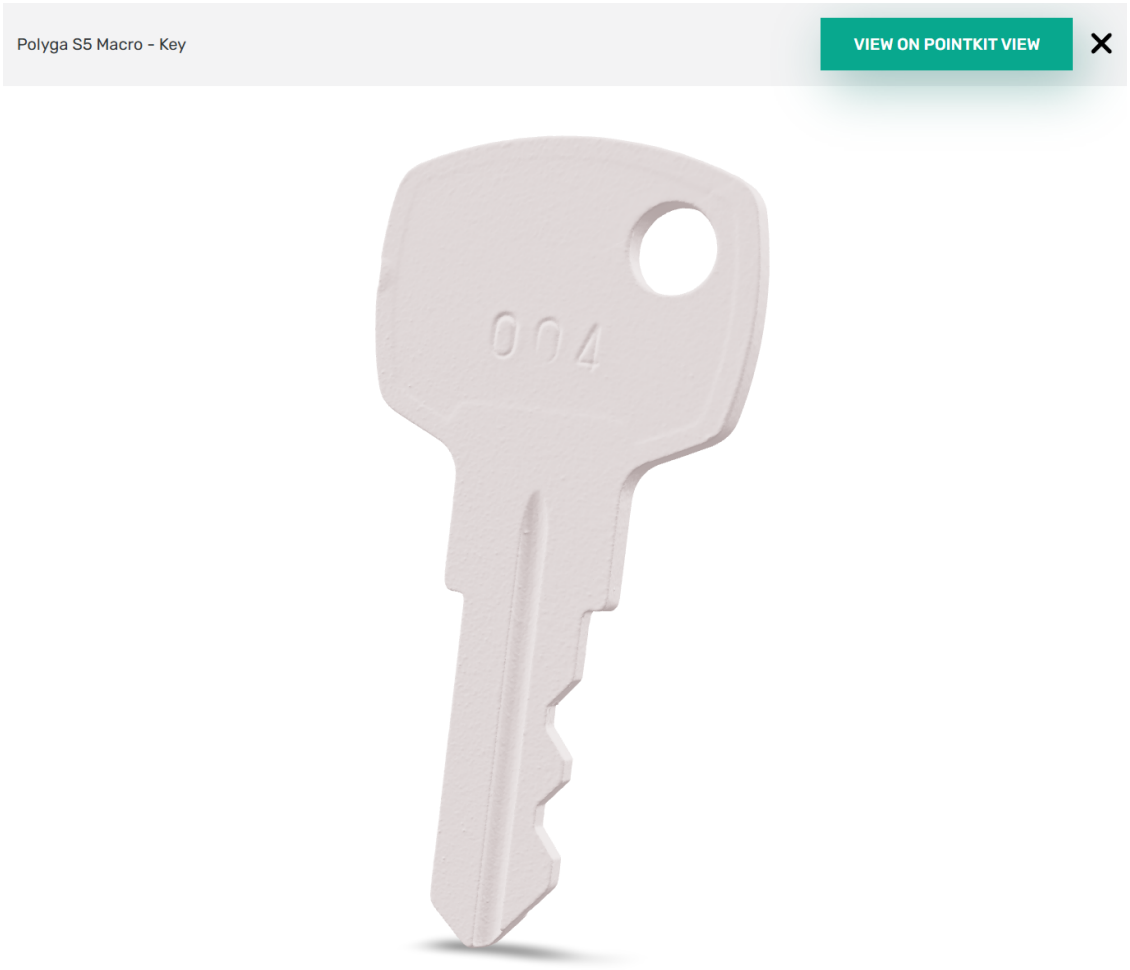
- Wierzchołki (vertices) - Punkty w przestrzeni $\mathbb{R}^3$ o współrzędnych (x, y, z).
- Krawędzie (edges) - Linie łączące punkty, definiują strukturę siatki.
- Powierzchnie (faces) - Wielokąty, które są zdefiniowane przez zbiór krawędzi i wierzchołków. Zwykle są to trójkąty lub rzadziej czworokąty.

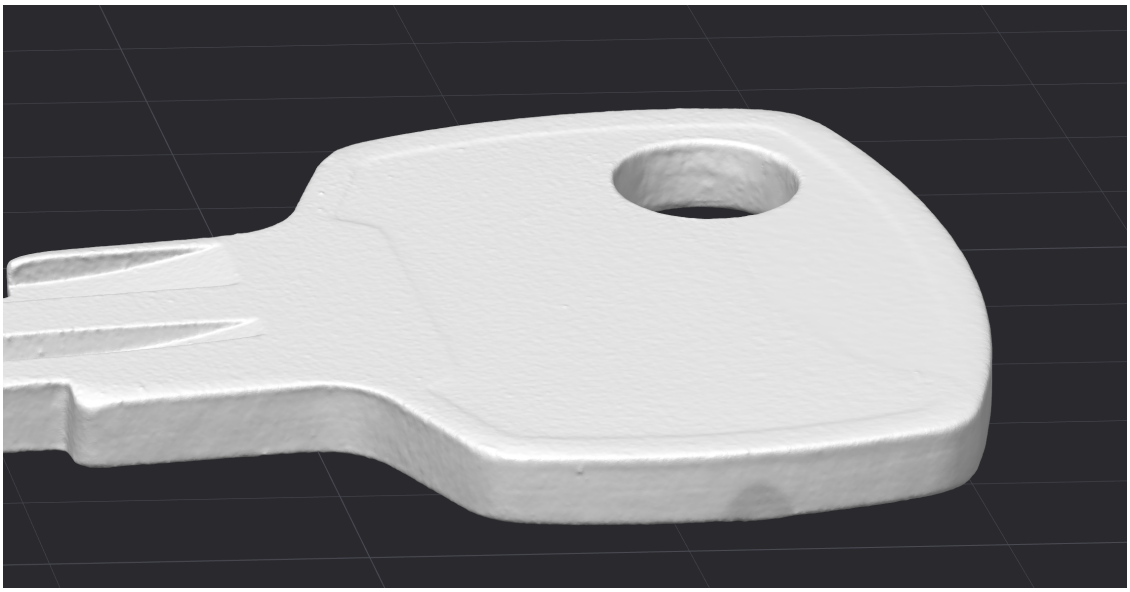
Istnieje wiele różnych algorytmów generowania siatek trójkątnych (czyli siatek, których powierzchnie są trójkątami), które różnią się przede wszystkim przyjmowanymi danymi wejściowymi. Siatkę trójkątną możemy wygenerować np. (omawianym na wykładzie) algorytmem Marching Cubes, który generuje siatkę trójkątną z trójwymiarowych danych skalarnych (pola skalarnego). Z kolei mając dane w formie chmury punktów, siatkę trójkątną wygenerować możemy np. za pomocą algorytmu Triangularyzacji Delaunaya.

Najczęściej w grafice komputerowej wykorzystuje się siatki trójkątne, głównie ze względu na ich "optymalność" i "praktyczność" (dokładniejsze omówienie tych określeń wymagałoby szczegółowego uzasadnienia i dowodów). Siatka trójkątna, choć może być bardzo dokładna, to nadal jest tylko dyskretną formą zapisu danych o powierzchni obiektu, co wiąże się z pewnymi ograniczeniami co do dokładności i anomaliami (artefaktami) w momencie, gdy chcemy uzyskać wysoką wydajność obliczeniową takiej siatki (np. na potrzeby real-time renderingu w grach). W tym przypadku przydatnym będzie zastosowanie algorytmu wygładzania siatki, który pozwoli nam na uzyskanie realistycznego obiektu 3D, który pomimo zawierania stosunkowo niedużej ilości trójkątów, będzie wysokiej jakości i nie będzie posiadać artefaktów.

3 Wygładzanie siatki (Mesh Smoothing)

W ramach przykładu, przyjmijmy, że posiadamy gotowy model 3D w formacie siatki trójkątnej (plik .stl), który otrzymaliśmy skanując rzeczywisty obiekt za pomocą skanera 3D. Pomimo posiadania wysokiej jakości skanera, ze względu na specyfikę działania tego typu urządzeń nie jesteśmy w stanie uzyskać dokładnego odwzorowania danego obiektu, przez co zeskanowany metalowy klucz posiada chropowatą strukturę (która jest de facto szumem), pomimo tego, że klucz w rzeczywistości jest gładki i błyszczący.





(Model należy do Polyga Inc., dostępny publicznie na https://www.polyga.com/w3/scan-samples/, grafiki własne)

Zatem chcąc uzyskać bardziej realistyczny model klucza, przydatne okaże się wygładzanie siatki 3D z wykorzystaniem operatora Laplace'a.

4 Wygładzanie Laplace'a (Laplacian Smoothing)

Na początku, wprowadźmy teorię dotyczącą operatora Laplace'a.

Operator Laplace'a: W przestrzeni $\mathbb{R}^3$, operator Laplace'a (laplasjan) Δ dla pola skalarnego $\phi(x, y, z)$ ma postać:

$$\Delta\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}$$

Znając już postać operatora Laplace'a, możemy wykorzystać go do rozwiązania równania różniczkowego postaci:

$$\frac{\partial X}{\partial t} = \lambda \cdot \Delta X$$

Gdzie $X = \{x_1, x_2, ..., x_n\}$ będzie zbiorem wierzchołków $x_i = (x, y, z)$ siatki naszego modelu, a $\lambda \in [0, 1]$ będzie skalarem, odpowiadającym za poziom wygładzania.
Dodatkowo, korzystając z następującej własności liniowości operatora Laplace'a:

$$\Delta(a\phi + b\psi) = a\Delta\phi + b\Delta\psi$$

Gdzie ϕ i ψ są dowolnymi funkcjami i $a, b \in \mathbb{R}$.
Możemy uzyskać następującą postać równania:

$$X(n+1) = X(n) + \lambda dt \Delta X(n)$$

Gdzie $X(n+1)$ to zbiór wierzchołków po kolejnym kroku wygładzania.

Po wyłączeniu $X(n)$ otrzymujemy równanie (1):

$$X(n+1) = (I + \lambda dt \Delta) X(n)$$

Gdzie I jest macierzą identycznościową.

Oczywiście, w algorytmie wygładzania korzysta się z zdyskretyzowanego operatora Laplace'a, wówczas (przybliżony) Laplasjan możemy zapisać w następujący sposób:

$$\Delta x * i = \sum *j \in N * 1(i) w * ij(x * j - x_i)$$

Gdzie x_i, x_j to wierzchołki siatki, $N_1(i)$ to najbliżsi sąsiedzi wierzchołka x_i, a $w * ij$ to wagi.
Wagi w_{ij} mogą być różne, zależne od pożądanego efektu, w podstawowej wersji mają postać $w_{ij} = \frac{1}{m}$, gdzie m to liczba sąsiadów wierzchołka x_i.

Wówczas równanie (1), które wykonujemy dla każdego wierzchołka $x_i \in X$ wygląda następująco:

$$x * i^{(n+1)} = x_i^{(n)} + \lambda \cdot \sum *j \in N * 1(i) w * ij(x_j^{(n)} - x_i^{(n)})$$

Należy zauważyć, że taka metoda wygładzania Laplace'a ma kilka przydatnych własności:

- Nie modyfikuje ona krawędzi siatki (nadal są one połączone do tych samych wierzchołków), a więc pierwotna triangularyzacja zostaje zachowana bez zmian.
- Do wygładzenia danego wierzchołka potrzeba jedynie danych jego sąsiednich wierzchołków, co pozwala na szybkie wygładzanie tylko wybranych miejsc siatki lub przeprowadzenie obliczeń równolegle (np. na GPU).
- Niska złożoność obliczeniowa rzędu $O(n)$.

5 Przykład zastosowania wygładzania Laplace'a

Wracając do wspomnianego wcześniej modelu klucza, opisane wyżej informacje o wygładzaniu Laplace'a możemy zastosować do wygładzenia naszego modelu klucza w celu uzyskania bardziej realistycznej, gładkiej powierzchni. W tym celu użyjemy gotowej funkcji `smooth()` z biblioteki PyVista w Pythonie (wykorzystuje ona domyślnie metodę Laplace'a, szczegóły w dokumentacji: https://docs.pyvista.org/api/core/_autosummary/pyvista.polydatafilters.smooth) .

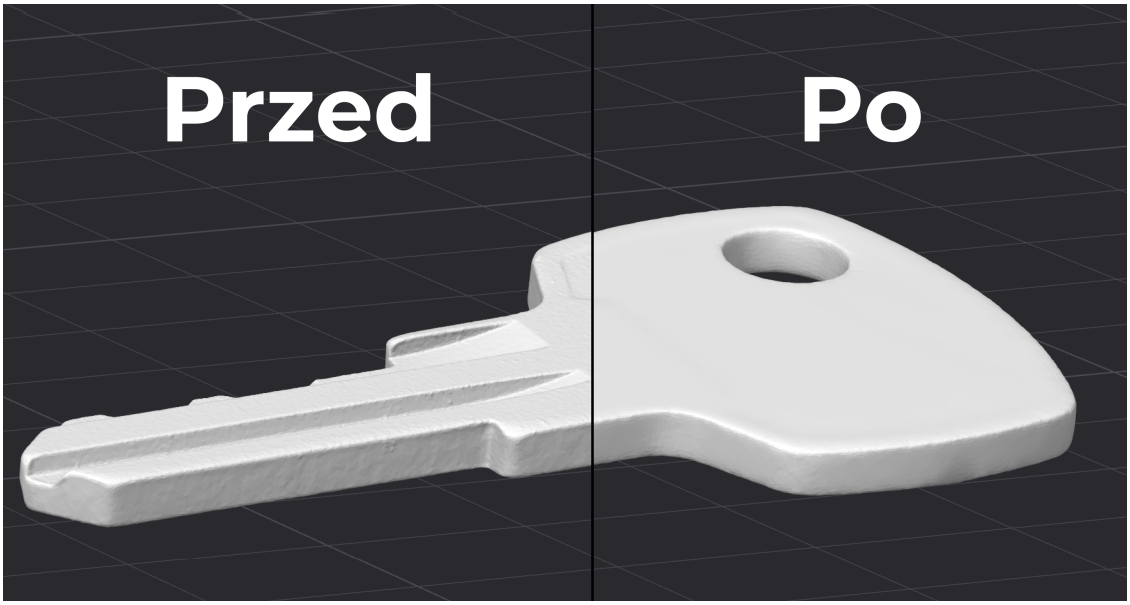
```
import pyvista as pv

mesh = pv.read('key.stl')

smooth_mesh = mesh.smooth(n_iter=50, relaxation_factor=0.1)

smooth_mesh.save('key_smooth.stl')
```

Powyższy kod pozwala na optymalne wygładzenie modelu klucza. Ustawiona liczba iteracji to 50, a λ (w bibliotece nazywana współczynnikiem relaksacji) wynosi 0.1.



Jak widać, matematyczna teoria ma również swoje praktyczne zastosowania w informatyce. Choć w przypadku biblioteki PyVista całość teorii dotyczącej m.in. operatora Laplace'a jest ukryta pod jedną funkcją, to znajomość podstaw matematycznych pozwala nam na zrozumienie jej działania i świadome jej zastosowanie.

6 Rozszerzenia algorytmu Laplace'a

Opiswany wyżej algorytm Laplace'a nie jest oczywiście algorytmem idealnym, który sprawdzi się w przypadku każdej siatki. Z tego powodu, powstają różne modyfikacje tego algorytmu, które minimalizują wady prostego algorytmu Laplace'a. Oto kilka z nich:

- Wygładzanie Taubina - Dodaje on ujemny parametr μ, który kontroluje stopień wygładzania "na zewnątrz" (przeciwstawnie do kierunku wygładzania Laplace'a), w celu ograniczenia efektu zmniejszania się objętości obiektu (innymi słowy, kurczenia się obiektu), tak jak ma to miejsce przy podstawowym wygładzaniu Laplace'a. Niestety wiąże się to z koniecznością znaczącego zwiększenia liczby iteracji.
- Skalowanie objętości - Uproszczone rozwiązanie wyżej wspomnianego problemu, polega ono na odpowiednim przeskalowaniu obiektu po wygładzeniu Laplace'a, tak aby wrócić do oryginalnej objętości.
- Modyfikacje wag - Użycie innych wag w_{ij} w algorytmie pozwala na zmniejszenie stopnia zniekształcenia obiektu o nieregularnych kształtach, pomocne w tym mogą być np. wagi Fujiwara.
- Paralelizacja obliczeń - Polega m.in. na modyfikacji struktury danych używanej w algorytmie Laplace'a, w celu wykonania szybszych obliczeń na jednostkach wielordzeniowych (np. na kartach graficznych NVIDIA z wykorzystaniem CUDA).

Pomimo wielu różnych modyfikacji algorytmu Laplace'a, każde z nich wiąże się z zaletami i wadami. Dlatego też ciągle powstają nowe metody wygładzania w celu znajdowania optymalnego położenia wierzchołków przy wygładzaniu siatki. Powstają również algorytmy wygładzania siatek wykorzystujące grafowe sieci neuronowe (https://arxiv.org/pdf/2311.12815v2), co jest dowodem na to, że jest jeszcze wiele do odkrycia w tym obszarze i konieczna jest do tego znajomość teorii matematycznej.

7 Źródła

Do napisania tego artykułu wykorzystano informacje z poniższych źródeł:

Materiały do wykładu z Analizy Matematycznej z Zastosowaniami 2
https://sound.eti.pg.gda.pl/student/so/01-Modelowanie.pdf
https://en.wikipedia.org/wiki/3D_modeling
https://paulbourke.net/geometry/polygonmesh/
https://www.pmp-book.org/download/slides/Smoothing.pdf
https://rucore.libraries.rutgers.edu/rutgers-lib/58677/PDF/1/play/
https://www.ljll.fr/~frey/papers/meshing/Bray%20N.,%20Notes%20on%20mesh%20smoothing.pdf
https://www.researchgate.net/publication/4228042_A_global_Laplacian_smoothing_approach_with_feature_
https://docs.pyvista.org/
https://arxiv.org/pdf/2311.12815v2