Detekcja żeber w modelach CAD

Projekt zespołowy – kierunek Projektowanie Systemów CAD/CAM

Zespół „Żebra"

Sebastian Szerafin
Emma Leveilley
Damian Tomczak

Wydział Matematyki i Nauk Informacyjnych
26 czerwca 2025

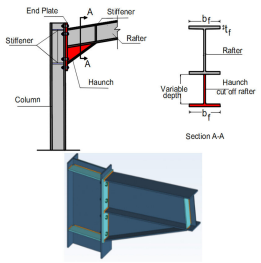
Spis treści

Rozdział 1

Opis problemu—detekcja żeber w modelach CAD

1.1 Czym są „żebra” (ang. *haunch*)?

Żebro to cienkościenny element wzmacniający lub usztywniający większą powierzchnię części – najczęściej spotykany w odlewach, elementach wtryskiwanych z tworzyw sztucznych oraz w konstrukcjach blaszanych. Typowa geometria żebra to *dwie równoległe płaszczyzny* oddalone o grubość d, połączone powierzchniami bocznymi (rys. 1.1). Parametry technologiczne (minimalna grubość, maksymalne pochylenie, promień zaokrąglenia u nasady) zależą od procesu wytwarzania.



Rysunek 1.1: Przykładowe żebro wzmacniające w części blaszanej.

1.2 Formalizacja problemu

1.2.1 Definicja geometryczna

W algorytmie przyjmujemy, że *żebro* stanowi para płaskich powierzchni F_1, F_2, które spełniają jednocześnie:

Równoległość płaszczyzn. Normale $\vec{n}_1, \vec{n}_2$ są równoległe w granicach tolerancji

$$\text{angle}(\vec{n}_1, \vec{n}_2) \leq \varepsilon_\alpha, \qquad \varepsilon_\alpha = \texttt{Precision::Angular()} \approx 10^{-7} \text{ rad.}$$

Stały odstęp. Średni dystans między płaszczyznami jest równy d, a dla każdego wierzchołka $v_1 \in F_1$ istnieje odpowiadający wierzchołek $v_2 \in F_2$ taki, że

$$\Big|\, |\,\text{proj}_{\vec{n}}(v_2 - v_1)| - d \Big| \;<\; \varepsilon_d, \qquad \varepsilon_d = 10^{-4} \text{ (jednostki modelu).}$$

Izomorfizm konturów. Po rzutowaniu obu zestawów wierzchołków na płaszczyznę F_1 każdemu punktowi v_1 można przyporządkować punkt v_2 spełniający warunek (b) oraz

$$\| \, v_2 - v_1 - \text{proj}_{\vec{n}}(v_2 - v_1) \| \;<\; \varepsilon_{xy}, \qquad \varepsilon_{xy} = 10^{-4}.$$

(Algorytm dopasowuje wierzchołki jako multisety; nie wymaga zgodności kolejności ani orientacji krawędzi.)

Rozdział 2

Podejścia do problemu

2.1 Podejście AI

2.1.1 Dlaczego nie działa?

W podejściu z wykorzystaniem uczeniem maszynowego pokładaliśmy duże nadzieje, niemniej aby uzyskać jakikolwiek sensowny wynik należałoby włożyć ogromną ilość czasu wykraczającą poza czas trwania przedmiotu. Wybróbowanie tego podejścia stworzyło świetny materiał pod dalsze badania tematu np. pod pracę dyplomową.
Przedewszystkim należy zauważyć, że nie dysponowaliśmy danymi treningowymi dla sieci, posiłkowaliśmy się jednym modelem domu z dwoma żebrami oznaczonymi i jednym nieoznaczonym, przedstawionych w formacie obj (kolejne żebra były przechowywane jako n-krotnione powielone wierzchołki), co jest absurdalne. Zasoby internetu nie oferują darmowymi zasobami treningowymi, a wytworzenie odpowiedniej ilości konstrukcji z żebrami pożarłoby ogromną ilość czasu roboczego i nauki/przypomnienia obsługi programów cad.
W podejściu AI można wyszczególnić dwa etapy, przed i po rozmowie z zaprzyjaźnionym studentem kierunku Metody sztucznej inteligencji, który pozwolił, zrozumieć, że dotychczasowe próby próbowały nauczyć sieć nie relacji pomiędzy wierzchołkami w żebrach, a sztywnym położeniem wierzchołków w przestrzeni. Wcześniejsze próby posiłkowały się pobieżnym researchem w internecie i rozmowami z modelem o3 ChatGPT. Poniższa rozprawa, opiera się właśnie o ostatnie najbardziej obiecujące podejście, przed uświadomieniem sobie ogromu pracy, którą należałoby włożyć aby uzyskać satysfakcjonujący wynik.

2.1.2 Przegląd technologii

Warstwa	Technologia	Rola w pipeline
Język / runtime	C++17	Wydajna implementacja K-NN i parsowania dużych plików OBJ.
Framework DL	LibTorch (C++ API PyTorch)	Definicja i trenowanie sieci, optymalizacja AdamW, serializacja modelu.
Format danych	Wavefront OBJ	Wejściowa geometria konstrukcji stalowych.
Alg. sąsiedztwa	Własna implementacja k-NN	Dostarcza lokalny kontekst przestrzenny (wektory Δ).
Sprzęt	CPU / CUDA GPU	Przyspieszenie uczenia i inferencji.

Tabela 2.1: Technologie wykorzystane w projekcie.

2.1.3 Struktura danych wejściowych

Wierzchołek bazowy surowe współrzędne $(x, y, z) \in \mathbb{R}^3$.

Kontekst k-NN Dla każdego punktu wyznaczamy k najbliższych sąsiadów

$$\Delta_i = (\Delta x_i, \Delta y_i, \Delta z_i) = p_{\text{nbr},i} - p_{\text{base}}.$$

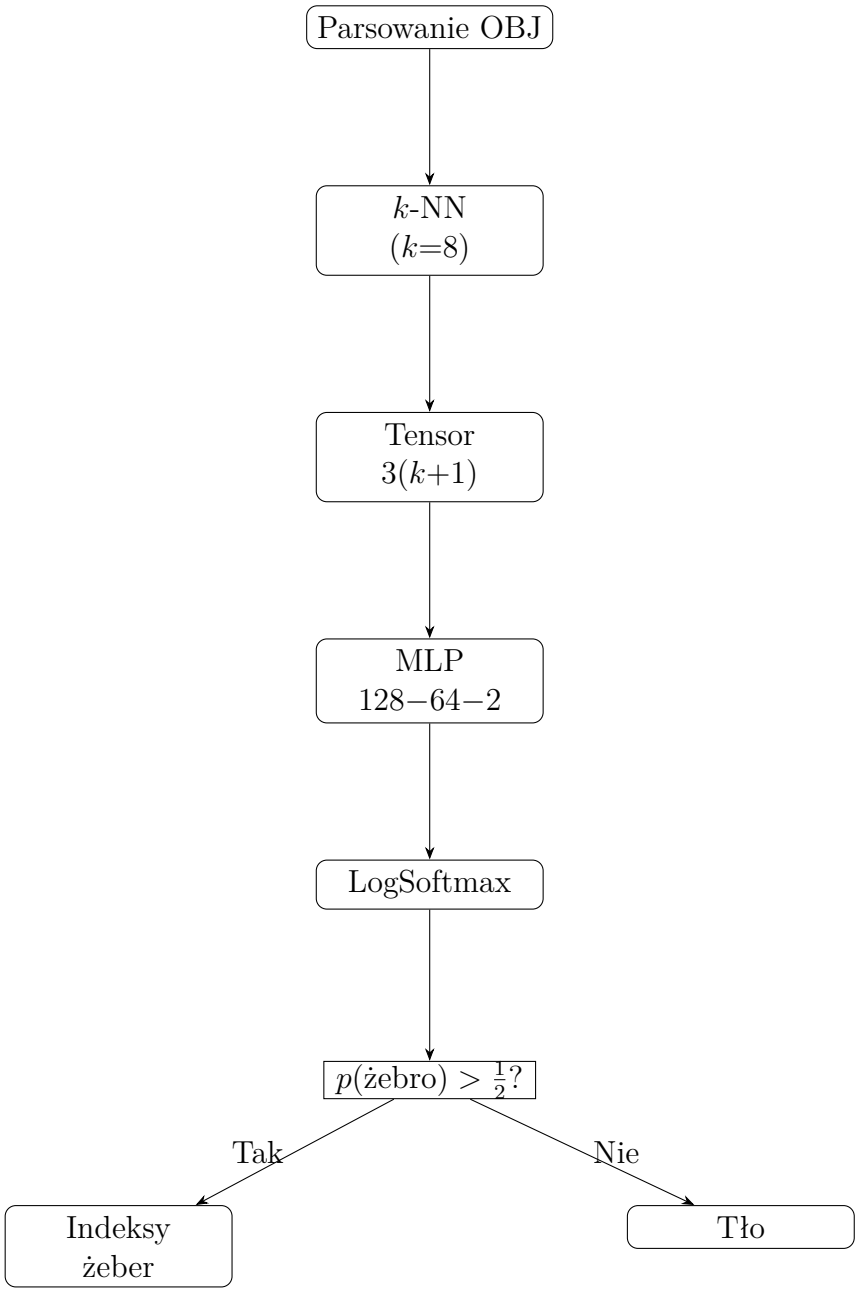
Wektor cech

$$\underbrace{\big[x, y, z, \Delta_1, \ldots, \Delta_k\big]}_{\in \mathbb{R}^{3(k+1)}}.$$

W kodzie przy $k = 8$ otrzymujemy 27-wymiarowy tensor typu `float32`.

2.1.4 Algorytm detekcji żeber

Schemat przepływu



Rysunek 2.1: Przepływ danych w pipeline wykrywania żeber.

2.1.5 Model

- **Wejście:** $3(k+1)$ cech (xyz + k wektorów Δ).
- **Warstwy:** `Linear`$\big(3(k+1), 128\big)$ $\to$ReLU$\to$ `Linear`$(128, 64)$ $\to$ReLU$\to$ `Linear`$(64, 2)$ $\to$ LogSoftmax.
- **Funkcja kosztu:**

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N} w_{y_n}\,\log p_\theta(y_n \mid \mathbf{x}_n), \quad w_c = \frac{N_{\text{total}}}{2\,N_c}.$$

- **Optymalizator:** AdamW ($\eta = 10^{-3}$).

2.2 Podejście analityczne

2.2.1 Założenia

Algorytm korzysta wyłącznie z informacji topologiczno–geometrycznej dostarczanej przez jądro Open Cascade Technology (OCCT). Przyjmujemy, że:

1. żebro tworzy *dwie* płaskie, wzajemnie równoległe ściany oddalone o grubość $d \le d_{\max}$,
2. kontury obu ścian są identyczne w rzutowaniu na płaszczyznę żebra (żaden wierzchołek nie wypada poza tolerancję τ),
3. możliwe są niewielkie odchyłki numeryczne ($\tau = 10^{-4}$ mm, $\varepsilon_\alpha = 0{,}5°$).

2.2.2 Algorytm krok po kroku

Ekstrakcja ścian płaskich: `CollectPlaneFaces` przeszukuje wszystkie `FACE` modelu i zachowuje jedynie te o typie `GeomAbs_Plane`.

Porównanie parami: Dla każdej nieuporządkowanej pary płaszczyzn (F_i, F_j) sprawdzamy równoległość normalnych (`gp_Dir::IsParallel`).

Pomiar odległości: punkt referencyjny $P_i = F_i(u{=}0, v{=}0)$, $P_j = F_j(0,0)$; $d_{ij} = |P_iP_j|$. Jeśli $d_{ij} \le d_{\max}$ przechodzimy dalej.

Weryfikacja izomorfizmu konturów: `HaveSameVertices` sprawdza, czy każdy wierzchołek $p \in \partial F_i$ ma odpowiadający mu wierzchołek $q \in \partial F_j$ przesunięty dokładnie o d_{ij} wzdłuż normalnej.

Oznaczenie wyniku: obie ściany, spełniające warunki 1–4, traktujemy jako powierzchnie żebra i kolorujemy na czerwono.

2.2.3 Kluczowa implementacja w C++

Listowanie 2.1 przedstawia główne funkcje; pełna wersja dostępna w pliku `haunch.cpp`.

```
bool GetFacePlaneNormal (const TopoDS_Face& face, gp_Dir& n)
{
  GeomAdaptor_Surface S(BRep_Tool::Surface(face));
  if (S.GetType() != GeomAbs_Plane) return false;
  n = Handle(Geom_Plane)::DownCast(BRep_Tool::Surface(face))
        ->Pln().Axis().Direction();
  return true;
}

bool HaveSameVertices(const TopoDS_Face& f1,
                      const TopoDS_Face& f2, float d)
{
  TopTools_IndexedMapOfShape v1, v2;
  TopExp::MapShapes(f1, TopAbs_VERTEX, v1);
  TopExp::MapShapes(f2, TopAbs_VERTEX, v2);
  if (v1.Extent()!=v2.Extent()) return false;

  gp_Dir n; if (!GetFacePlaneNormal(f1,n)) return false;
  for (int i=1;i<=v1.Extent();++i)
  {
    gp_Pnt p = BRep_Tool::Pnt(TopoDS::Vertex(v1(i)));
    bool ok=false;
    for (int j=1;j<=v2.Extent();++j)
    {
      gp_Pnt q = BRep_Tool::Pnt(TopoDS::Vertex(v2(j)));
      gp_Vec dPQ(p,q);
      double h = std::abs(dPQ.Dot(gp_Vec(n))); // osiowo
      gp_Vec lateral = dPQ - h*gp_Vec(n); // poprzecznie
      if (lateral.Magnitude()<1e-4 && std::abs(h-d)<1e-4){ok=true;break;}
    }
    if (!ok) return false;
  }
  return true;
}
```

Listing 2.1: Detekcja par płaszczyzn – fragment podejścia analitycznego

2.2.4 Zalety i ograniczenia

Plusy

- deterministyczny, w 99 % powtarzalny wynik,
- brak potrzeby danych uczących,
- minimalne zależności (czysty OCCT, brak GPU).

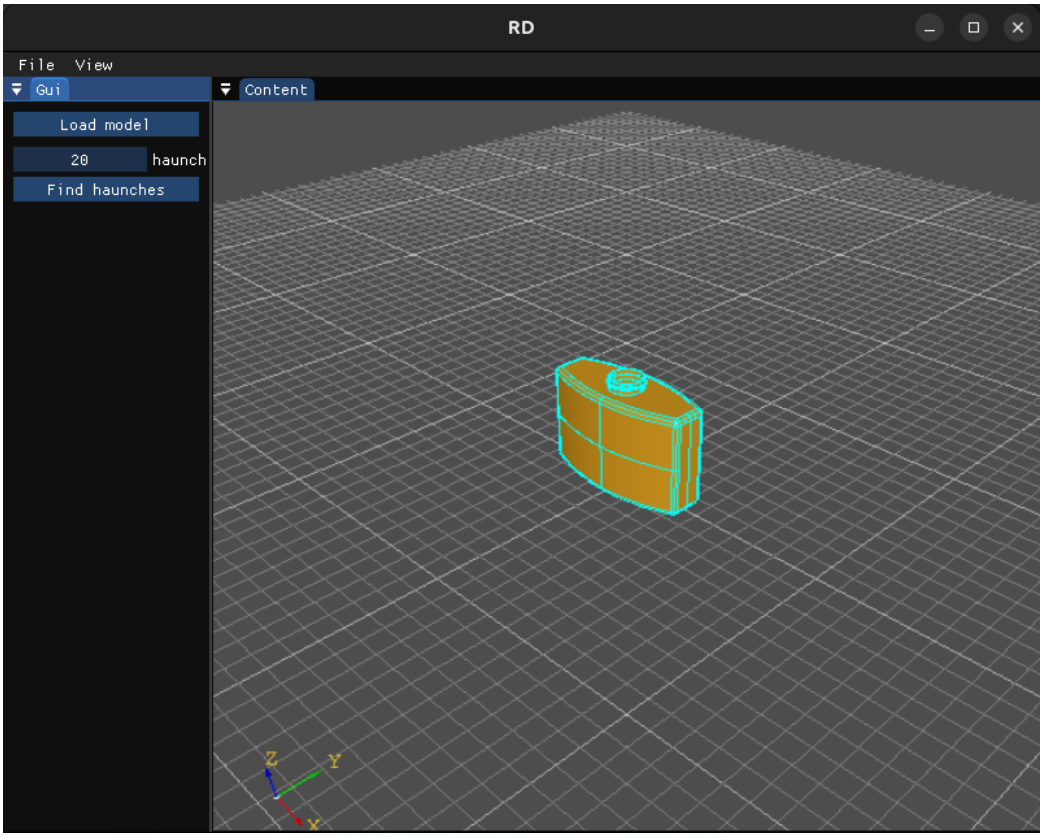
Minusy

- wrażliwość na zaokrąglenia i lekkie skrzywienia powierzchni;

- złożoność kwadratowa w M (istotna przy bardzo złożonych modelach blaszanych);
- nie radzi sobie z żebrami o przekroju łukowym lub podwójnie zagiętym.

Rozdział 3

Opis aplikacji



Rysunek 3.1: Interfejs aplikacji

Program stanowi aplikacja okienkowa z graficznym interfejsem użytkownika, umożliwiająca analizę modeli CAD w celu automatycznego wykrywania żeber przy użyciu biblioteki Open Cascade. Po uruchomieniu użytkownikowi

prezentowana jest pusta scena 3D, stanowiąca przestrzeń roboczą, w której mogą być wizualizowane wczytane modele.

Interfejs graficzny, choć uproszczony, zapewnia wszystkie niezbędne funkcje potrzebne do przeprowadzenia pełnej analizy geometrii w kontekście detekcji żeber. Aplikacja pozwala na wykonanie następujących operacji:

- **Wczytanie modelu 3D** – użytkownik może zaimportować model w formacie BREP, który zostaje następnie wyświetlony na scenie,
- **Parametryzacja** – możliwe jest określenie dopuszczalnej grubości elementów, które mają zostać zakwalifikowane jako żebra,
- **Uruchomienie algorytmu detekcji** – na podstawie wprowadzonych parametrów aplikacja analizuje topologię i geometrię modelu w celu automatycznego wykrycia i wyróżnienia żeber konstrukcyjnych.

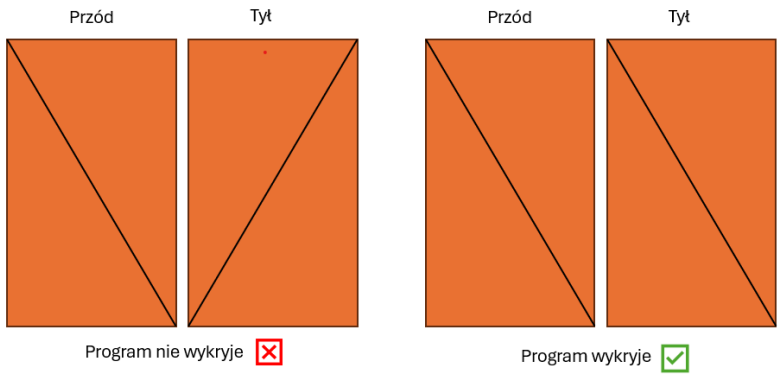
Działanie aplikacji opiera się na wykorzystaniu funkcjonalności OCCT do reprezentacji, przetwarzania i wizualizacji geometrii 3D. Wyniki działania algorytmu (znalezione żebra) są prezentowane bezpośrednio na scenie graficznej, poprzez podświetlenie na czerwono, co pozwala na ich szybkie zlokalizowanie.

Rozdział 4

Model domku i komplikacje

Podczas pracy nad projektem napotkano brak dostępnych modeli w formacie BREP zawierających żebra, niezbędne do testowania ich wykrywania. Znaleziony model 3D domku zawierał żebra, ale zapisany był w formacie .obj.

Konwersja tego modelu do formatu BREP nie pozwoliła na poprawne wykrycie żeber, ponieważ podczas triangulacji program nie traktował przeciwnych płaszczyzn jako odpowiadających sobie (Rysunek 4.1, lewa strona).



Rysunek 4.1: Triangulacja żeber po konwersji do brepa.

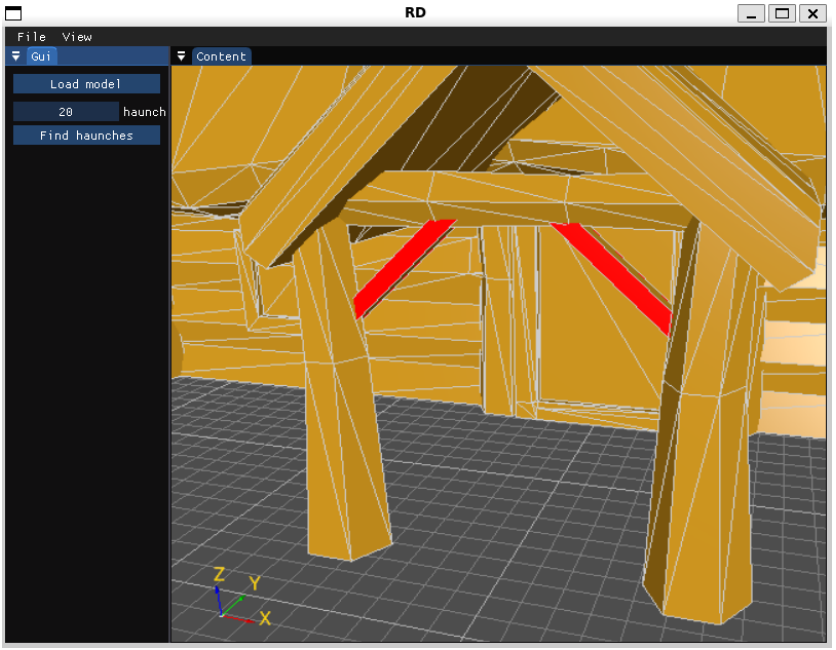
Nawet eksportowanie modelu z quadami nie rozwiązało problemu, ponieważ FreeCAD automatycznie triangulował powierzchnie podczas tworzenia BREP, nie zachowując odpowiednio definicji żeber (z Rozdziału 1.2.1).

Ostatecznie rozwiązano problem przez usunięcie w programie Blender istniejących żeber w modelu, wstawienie w ich miejsce równoległościanów, a następnie osobne załadowanie modelu i żeber w FreeCAD. Żebra wczytano za pomocą skryptu Pythona, co umożliwiło eksport do BREP bez triangulacji.

Rozdział 5

Rezultaty

Po przygotowaniu poprawnego modelu, udało się zaimportować go do aplikacji i skutecznie wykryć wszystkie żebra zgodnie z przyjętą definicją. Finalny efekt działania programu przedstawiono na Rysunku 5.1.



Rysunek 5.1: Wynik aplikacji.

Rozdział 6

Subiektywna opinia o bibliotece Open Cascade

Biblioteka Open Cascade oferuje bardzo szeroki zakres funkcjonalności do pracy z geometrią bryłową i topologią modeli 3D, co czyni ją potężnym narzędziem w projektach inżynierskich i CAD-owych. Jej zaletą jest otwartość, elastyczność i bogaty zestaw narzędzi do analizy oraz przetwarzania geometrii. Z drugiej strony, dokumentacja bywa niekompletna, a próg wejścia dla nowych użytkowników jest stosunkowo wysoki – wiele funkcji wymaga głębokiego zrozumienia struktury danych i wewnętrznego działania biblioteki. Mimo to, OCCT pozostaje wartościowym rozwiązaniem dla bardziej zaawansowanych projektów, w których konieczne jest pełne panowanie nad geometrią 3D i jej interpretacją.