

6. TRIANGULACJA DELAUNAY

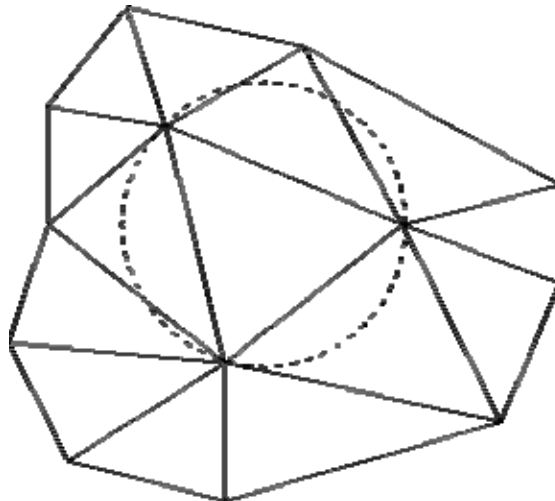
Niniejsze ćwiczenie ma na celu zapoznanie studentów z metodami podziału obszaru na trójkąty. Mając do dyspozycji punkty wejściowe obszaru należy je w ten sposób przetworzyć, aby w pliku wyjściowym znalazły się wierzchołki trójkątów.

Wprowadzenie teoretyczne

Definicja

Triangulacja Delaunay to taki podział obszaru na trójkąty, gdzie każde koło opisane na trzech sąsiadujących ze sobą punktach z tesalacji Voronoia nie zawiera żadnego innego punktu (patrz rysunek 1).

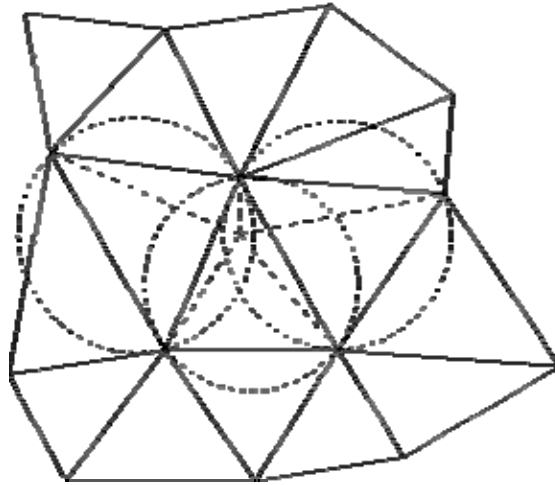
Rys 1: Ilustracja triangulacji Delaunay



Algorytm Bowyera-Watsona (B-W)

Algorytm ten określa sposób triangulacji Delaunay po dodaniu nowego punktu tak, aby nowa triangulacja spełniała również warunki Delaunay. Po wstawieniu nowego punktu trzeba sprawdzić, do których okręgów opisanych on należy i połączyć go ze wszystkimi wierzchołkami trójkątów, na których były one opisane. W ten sposób powstanie nowa triangulacja Delaunay. Ten zupełnie podstawowy algorytm zilustrowany jest na rysunku 2.

Rys 2: Ilustracja algorytmu Bowyera-Watsona



Algorytm Greena-Sibsona (G-S)

Jest to poprawiony algorytm Bowyera-Watsona, wymyślony w tym samym celu, czyli w celu wstawiania nowych punktów do triangulacji Delaunay. Po dodaniu nowego punktu w obszarze, najpierw łączymy go z wierzchołkami trójkąta do którego należy, a dopiero później sprawdzamy, czy okręgi opisane na powstałych w ten sposób trójkątach nie zawierają innych punktów. Jeśli zawierają, to zmieniamy połączenia między punktami podobnie jak w algorytmie B-W. Różnica w stosunku do algorytmu Bowyera-Watsona nie jest wielka, ale G-S jest zdecydowanie szybszy, bowiem nie zawsze konieczne jest sprawdzanie wszystkich trójkątów.

Algorytm Tenemura-Merriana (T-M)

Algorytm ten określa postępowanie przy dodawaniu nowych trójkątów brzegowych w metodzie postępującego brzegu (ang. Advancing Front Method -- AFM). Aby zdecydować, który z punktów należy połączyć z punktami na brzegu (tworząc w ten sposób nowy trójkąt), sprawdza się interakcyjnie, czy dla kolejnych prób są „puste okręgi opisane”. Jest używany przy generacji siatek z pomocą metody Delaunay wstawiania punktów.

Triangulacja Delaunay z ograniczeniami

Jest to szczególny rodzaj triangulacji Delaunay. Dotyczy obszarów składających się z dwóch lub więcej części (domen). Granica oddzielająca te podobszary nie może ulec przesunięciu, ani przerwaniu. Dlatego cały obszar nie musi spełniać warunków Delaunay. Muszą one być jednak spełnione dla każdego podobszary z osobna. Jest to rozszerzenie definicji triangulacji Delaunay w celu dostosowania jej do warunków fizycznych. Różne podobszary oznaczają najczęściej różne materiały fizyczne.

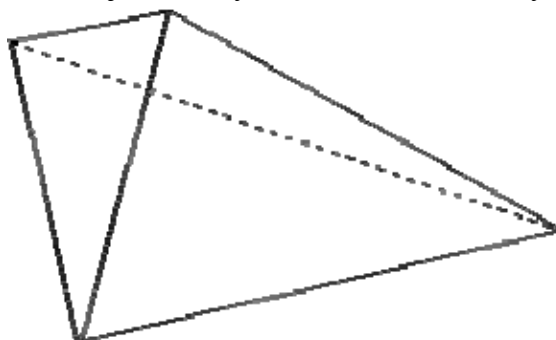
Niestety można udowodnić, że nie da się sformułować odpowiednika tej definicji w trzech wymiarach. Z tej przyczyny budowanie siatki trójwymiarowej, która ma zawierać określone siatki powierzchniowe (jest tak prawie zawsze) jest trudnym problemem.

W celu poprawy istniejącej już triangulacji, zbliżeniu się do optimum globalnego, stosuje się różne techniki zmiany konfiguracji sieci.

Dwuwymiarowa zamiana krawędzi

opiera się na tym, że istnieją dwa sposoby triangulacji czterech punktów, tak jak to pokazano na rysunku 3. W celu wybrania korzystniejszego układu można stosować różne optymalizacje, na przykład maksymalizację minimalnego kąta.

Rys 3: Ilustracja zamiany krawędzi w dwóch wymiarach



Algorytm przecinających się krawędzi

to metoda podobna do dwuwymiarowej zamiany krawędzi, ale umożliwiająca optymalizację bardziej globalną, a nie tylko w czterech punktach.

Trójwymiarowa zamiana ścian

to technika oparta na twierdzeniu sformułowanym przez Lawsona, że w przestrzeni d wymiarowej, z $d+2$ punktów można utworzyć elementy na co najwyżej dwa sposoby. Z pięciu punktów w trzech wymiarach mogą powstać 2, 3 lub 4 czworościany. W każdym z tych przypadków stosuje się inny mechanizm zamiany ścian.

Polecenia:

1. Implementacja algorytmu triangulacji Delauney'a w Matlabie

Dokonać implementacji algorytmu triangulacji Delauney'a w Matlabie według definicji algorytmów przedstawionych powyżej lub opisu zawartego poniżej lub wg własnego zaproponowanego algorytmu. Przygotować tak kod źródłowy programu, aby plik wynikowy można było prezentować w programie 'gnuplot'. Można się oprzeć na kodzie programu załączonym w dalszej części ćwiczenia.

Przykładowy algorytm.

1. Wśród podanych punktów poszukaj dwóch najbliższych położonych i traktuj je jako pierwszą prostą.
2. Pobierz trzeci punkt różny od tych dwóch łącząc je wyznacz pierwszy trójkąt.
3. Opisz okrąg na trójkącie tzn. znajdź jego środek oraz promień.
4. Sprawdź czy wewnątrz okręgu znajduje się jakiś punkt przez wyliczenie długości odcinka między danym punktem a środkiem okręgu.
 - Jeżeli długość jest mniejsza od promienia - obrany punkt leży wewnątrz okręgu i przyjmij go jako nowy wierzchołek trójkąta - wyznacz nowy środek okręgu i promień następnie skok do pkt.4
 - Jeżeli nie ma punktów, dla których odcinek łączący dany punkt i środek okręgu jest mniejszy od promienia - skok do pkt. 5.
5. Zaznacz trójkąt określony wg. powyższych kryteriów.
6. Sprawdź czy istnieją nowo utworzone trójkąty:

6.1. Jeżeli tak to pobierz kolejno utworzone trójkąty w celu tworzenia kolejnych trójkątów przylegających do już utworzonych poprzez:

- I. Z trójkąta wybierz dwa punkty i sprawdź po której stronie od tych punktów leży wierzchołek.
- II. Sprawdź czy istnieją punkty (na półpłaszczyźnie wyznaczonej przez prostą - po przeciwnej stronie od wierzchołka) z którymi można utworzyć trójkąt
 - Jeżeli są punkty pobierz taki punkt wyznacz środek okręgu opisanego na trójkącie i jego promień i skok do pkt. III.
 - Jeżeli nie ma wolnych punktów :
 - Wybierz kolejne dwa punkty w trójkącie i skok do pkt. II
 - Jeżeli brak punktów do sprawdzenia w trójkącie skok do pkt. 6.
- III. Sprawdź czy wewnątrz okręgu znajduje się jakiś punkt przez wyliczenie długości odcinka między danym punktem a środkiem okręgu.
 - Jeżeli długość jest mniejsza od promienia - obrany punkt leży wewnątrz okręgu i przyjmij go jako nowy wierzchołek trójkąta, wyznaczam nowy środek okręgu i promień następnie skok do pkt. III.
 - Jeżeli nie ma punktów, dla których odcinek łączący dany punkt i środek okręgu jest mniejszy od promienia - skok do pkt. IV.
- IV. Zaznacz wyznaczony trójkąt.
- V. Sprawdź czy :
 - Istnieją punkty w pobranym trójkącie na podstawie których nie było sprawdzane istnienie punktów do utworzenia trójkąta - pobierz je i skok do II.
 - Jeśli nie ma już takich punktów w pobranym trójkącie to skok do 6.

6.2. Jeżeli nie ma skok do pkt. 7.

7. Zakończenie Triangulacji.

Działanie algorytmu opiera się na pobieraniu punktów według kolejności w jakiej zostały wczytane z pliku. Powoduje to że w przypadku kilku punktów znajdujących się na badanym okręgu pobierany jest punkt który znajdował się najwcześniej na liście. W przypadku wyboru innego punktu z okręgu możliwe jest osiągnięcie innej siatki triangulacyjnej w okolicach tych punktów. W przypadku innych punktów sposób pobierania punktów nie ma wpływu na przeprowadzenie triangulacji.

1. Implementacja algorytmu triangulacji Delauney'a w Matlabie

Poniżej przedstawiono kod programu w Matlabie realizującego triangulację Delauney'a według algorytmu przedstawionego w instrukcji do ćwiczenia (kod programu w pliku triangulacja.m):

```
fid = fopen('punkty.txt');

if fid~-=-1
    points = fscanf(fid,'%g %g',[2 inf]);
    points = points';
    l_pkt = size(points,1);
    fclose(fid);
else
    l_pkt = 0;
end

plot(points(:,1),points(:,2),'o');

if l_pkt>2

    %siatka trójkątów triangulacyjnych:
    siatka_3k=[];

    %lista utworzonych trojkątów, do których można dobudować wierzchołki:
    list_3k=[];

    %najbliższe punkty:
    min_dist=inf;
    min_dist1=-1;
    min_dist2=-1;
    for i=1:l_pkt
        for j=i+1:l_pkt
            dist=odleglosc(points(i,1),points(i,2),points(j,1),points(j,2));
            if dist<min_dist
                min_dist=dist;
                min_dist1=i;
                min_dist2=j;
            end
        end
    end

    %kandydat na 3 wierzchołek:
    pt=-1;
    for i=1:l_pkt
        if i~min_dist1 && i~min_dist2 &&
~wspolliniowe(points(min_dist1,1),points(min_dist1,2),points(min_dist2,1),points(min_dist2,2),
points(i,1),points(i,2))
            pt=i;
            break;
        end
    end
```

```

end

%wybór trzeciego wierzchołka:
while 1
    %obliczenie środka i promienia okręgu dla pt:
    [xo_pt, yo_pt,
r2_pt]=okrag(points(min_dist1,1),points(min_dist1,2),points(min_dist2,1),points(min_dist2,2),p
oints(pt,1),points(pt,2));

    %sprawdzenie czy wewnątrz okręgu są inne punkty:
    no_pt=1;
    for i=1:l_pkt
        if i~=min_dist1 && i~=min_dist2 && i~=pt && odleglosc(xo_pt, yo_pt,
points(i,1), points(i,2))<r2_pt
            pt=i;
            no_pt=0;
            break;
        end
    end

    if no_pt
        break;
    end

end

%wstawiamy pierwszy trójkąt:
[list_3k, siatka_3k]=wstaw3k(min_dist1,min_dist2,pt,list_3k,siatka_3k,3);

%sprawdzanie kolejnych boków tworzonych trójkątów:
while size(list_3k,1)

    %kandydat na nowy wierzchołek:
    pt=-1;
    for i=1:l_pkt
        if i~=list_3k(1,1) && i~=list_3k(1,2) && i~=list_3k(1,3) &&
wpolplaszczynie(points(list_3k(1,1),1),points(list_3k(1,1),2),points(list_3k(1,2),1),points(l
ist_3k(1,2),2),points(list_3k(1,3),1),points(list_3k(1,3),2),points(i,1),points(i,2))
            pt=i;
            break;
        end
    end

    %wyznaczenie nowego wierzchołka:
    if pt>0
        while 1
            %obliczenie środka i promienia okręgu dla pt:
            [xo_pt, yo_pt,
r2_pt]=okrag(points(list_3k(1,1),1),points(list_3k(1,1),2),points(list_3k(1,2),1),points(list_
3k(1,2),2),points(pt,1),points(pt,2));

            %sprawdzenie czy wewnątrz okręgu są inne punkty:
            no_pt=1;
            for i=1:l_pkt
                if i~=list_3k(1,1) && i~=list_3k(1,2) && i~=list_3k(1,3) && i~=pt &&
odleglosc(xo_pt, yo_pt, points(i,1), points(i,2))<r2_pt
                    pt=i;
                    no_pt=0;
                    break;
                end
            end

            if no_pt
                break;
            end

        end

        %wstawienie nowego trójkąta na listę:
        [list_3k,
siatka_3k]=wstaw3k(list_3k(1,1),list_3k(1,2),pt,list_3k,siatka_3k,2);

    end

    %usunięcie sprawdzonego boku z listy:
    list_3k=list_3k(2:size(list_3k,1),:);

end

```

```

%rysowanie siatki trójkątów:
hold on;

for i=1:size(siatka_3k,1)
    plot([points(siatka_3k(i,1),1) points(siatka_3k(i,2),1) points(siatka_3k(i,3),1)
points(siatka_3k(i,1),1)], [points(siatka_3k(i,1),2), points(siatka_3k(i,2),2)
points(siatka_3k(i,3),2) points(siatka_3k(i,1),2)]);
    end

hold off;

%zapis do pliku:
fid=fopen('siatka.txt','w');
if fid~-=-1
    for i=1:size(siatka_3k,1)
        fprintf(fid,'%g %g\n%g %g\n%g %g\n%g %g\n\n',points(siatka_3k(i,1),1),
points(siatka_3k(i,1),2), points(siatka_3k(i,2),1), points(siatka_3k(i,2),2),
points(siatka_3k(i,3),1), points(siatka_3k(i,3),2), points(siatka_3k(i,1),1),
points(siatka_3k(i,1),2));
    end
    fclose(fid);
end

end

```

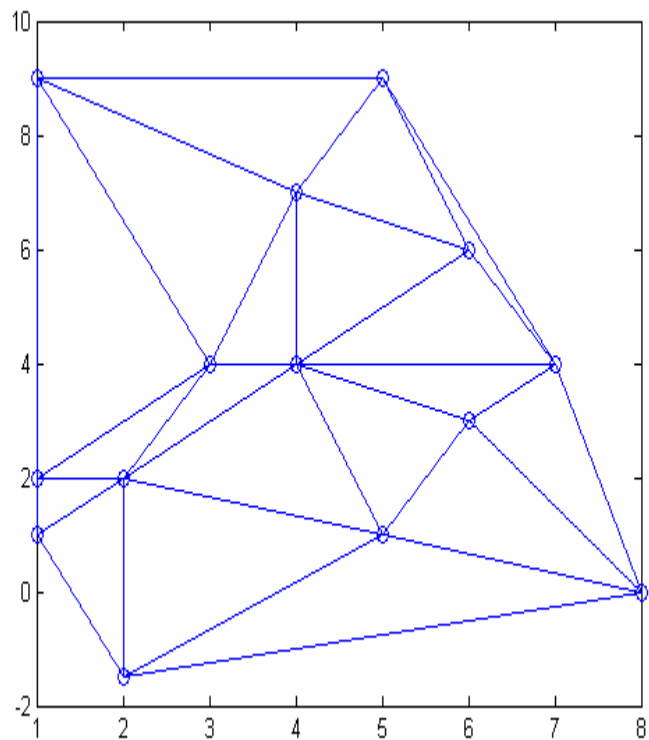
Program dokonuje triangulacji dla punktów o współrzędnych wczytywanych z pliku o nazwie „punkty.txt”. Wyniki triangulacji – współrzędne wierzchołków z powtórzonymi współrzędnymi początkowego wierzchołka zapisywane są do pliku „siatka.txt”, co umożliwia wykorzystanie tych danych do narysowania siatki z użyciem innego narzędzia.

Dodatkowo, program wyświetla obliczoną siatkę triangulacyjną oraz punkty na wykresie.

Funkcje wymagane do wykonania programu zaimplementowano w osobnych plikach:

- odleglosc.m
- okrag.m
- prosta_prostopadla.m
- punkt_przeciecia.m
- rownanie_prostej.m
- symetralna.m
- wpolplaszczyznie.m
- wspaniowe.m
- wstaw3k.m

Wynik triangulacji dla przykładowego zestawu punktów (rys. 1):



Rys. 1. Przykład działania programu. Punkty zaznaczono okręgami.

Do sprawozdania proszę dołączyć kod źródłowy w Matlabie oraz odpowiedzi na zadane pytania. W kodzie źródłowym proszę umieścić informację o autorstwie kodu.

Uwagi organizacyjne:

1. Katalogiem roboczym do umieszczania własnych plików jest `x:\`. Przed przystąpieniem do ćwiczeń należy usunąć wszystkie pliki z tego katalogu.
 2. Kod źródłowy w wersji elektronicznej znajduje się w katalogu `v:\biometrics\Delaunay`.
-