LAMPIRAN

```
%Lampiran Program Algoritma Pendeteksian Alur Pahatdan Orientasinya
function A=MainProgram()

banyakAnimasi = 10;
lebarBucket = 5;
tinggiBucket = 5;
radiusTool = 2;
panjangTool = 40;
kemiringan = 30;
heightTolerance = 5;   % toleransi perbedaan tinggi untuk cekungan
zTolerance = 3; % toleransi ketinggian untuk membuat garis melintang
r=radiusTool;
fid=fopen('gambar9.txt');
%tampilkanGambar(fid);
xMinMax = [1000 -1000];
yMinMax = [1000 -1000];
zMinMax = [1000 -1000];
i=1;
j=1;
count=0;
arr = zeros(3);
while (1)
   aa=fgetl(fid);
   if ~ischar(aa),
      break;
   end
   [a,b] = strtok(aa,' ');
   if strcmp(a,'facet')
      [c,d]= strtok(b,' ');
      for k=4:6,
         [e,d] = strtok(d,' ');
         hihi = str2double(e);
         T(i,k) = hihi(1);
      end
      % untuk penggambaran
      ii = 0;
      % end untuk penggambaran
      count=1;
   elseif strcmp(a,'endsolid'),
      i=i-1;
      j=j-1;
   elseif strcmp(a,'endfacet')
      T(i,1)=arr(1);
      T(i,2)=arr(2);
      T(i,3)=arr(3);
      i=i+1;
      % end untuk penggambaran
   elseif strcmp(a,'vertex')
      % untuk penggambaran
      % pembacaan X
      ii=ii+1;
      % end untuk penggambaran

      [a,b]= strtok(b,' ');
      [b,c]= strtok(b,' ');
      haha=str2double(a);
```

```
      huhu(1) = haha;
      X(ii) = haha(1);
      haha=str2double(b);
      huhu(2) = haha;
      Y(ii) = haha(1);
      haha=str2double(c);
      huhu(3) = haha;
      Z(ii) = haha(1);

        V(j,1)=0;
        V(j,2)=0;
        V(j,3)=0;
        %disp([V(j,1) V(j,2) V(j,3)])

      index=0;
      for kaka= 1:j,
        if (abs(V(kaka,1)-huhu(1))<0.001 && abs(huhu(2)-V(kaka,2))<0.001 && abs(huhu(3)-
V(kaka,3))<0.001)
          index=kaka;
          break;
        end
      end
      if index==0, % masukkan ke vertex

          %disp('masuk');
        V(j,1)=huhu(1);
        V(j,2)=huhu(2);
        V(j,3)=huhu(3);
        if xMinMax(1)>V(j,1),
          xMinMax(1)=V(j,1);
        end
        if xMinMax(2)<V(j,1),
          xMinMax(2)=V(j,1);
        end
        if yMinMax(1)>V(j,2),
          yMinMax(1)=V(j,2);
        end
        if yMinMax(2)<V(j,2),
          yMinMax(2)=V(j,2);
        end
        if zMinMax(1)>V(j,3),
          zMinMax(1)=V(j,3);
        end
        if zMinMax(2)<V(j,3),
          zMinMax(2)=V(j,3);
        end
        arr(count)=j;
      else
        arr(count)=index;
          %disp('tidak masuk');
        j=j-1;
      end
      j=j+1;
      count=count+1;
    end
end
```

```
% normalization
% calculate the distance from left-bottom-front of the model (minimum of
% the bounding box)
deltaX=xMinMax(1)-0;
deltaY=yMinMax(1)-0;
deltaZ=zMinMax(2)-0;

% calculate the bounding box to be placed in the normal position
xMinMax(1)=xMinMax(1)-deltaX;
xMinMax(2)=xMinMax(2)-deltaX;
yMinMax(1)=yMinMax(1)-deltaY;
yMinMax(2)=yMinMax(2)-deltaY;
zMinMax(1)=zMinMax(1)-deltaZ;
zMinMax(2)=zMinMax(2)-deltaZ;
% move all the vertex to the normal position
for ii=1:j,
   V(ii,1)=V(ii,1)-deltaX;
   V(ii,2)=V(ii,2)-deltaY;
   V(ii,3)=V(ii,3)-deltaZ;
   %disp([ii V(ii,1) V(ii,2) V(ii,3)]);
end
% end of normalization

%buketing
jmlBuketX=(xMinMax(2)-xMinMax(1))/lebarBucket; % see how many bucket in x position based on the
bounding box and bucket width
jmlBuketY=(yMinMax(2)-yMinMax(1))/tinggiBucket; % see how many bucket in y position based on the
bounding box and bucket height
jmlBuketX=floor(jmlBuketX)+1;  % to make sure that the numbers of bucket can handle all the triangles
and vertices
jmlBuketY=floor(jmlBuketY)+1;  % to make sure that the numbers of bucket can handle all the triangles
and vertices
jmlSegitiga = i;   % save the number of triangles into a variable "jmlSegitiga"
jmlVertex = j;     % save the number of vertices into a variable "jmlVertex"
buket = zeros(jmlBuketX,jmlBuketY);    % menyimpan jml bucket dalam setiap bucket (karena tidak ada
kelas vector seperti di java)
isiBuket = zeros(jmlBuketX,jmlBuketY); % menyimpan index-index segitiga dalam setiap bucket

% mulai penyimpanan segitiga
% dilakukan dengan cara mengecek
for i=1:jmlSegitiga,
   xBound = [1000 -1000];
   yBound = [1000 -1000];
   xv = zeros(3);
   yv = zeros(3);
   for j=1:3,
      Ver(1)=V(T(i,j),1); % x
      Ver(2)=V(T(i,j),2); % y

      posX = floor(Ver(1)/lebarBucket)+1;
      posY = floor(Ver(2)/tinggiBucket)+1;
      xv(j) = posX;
      yv(j) = posY;
      if xBound(1)>xv(j),
         xBound(1)=xv(j);
```

```
    end
    if xBound(2)<xv(j),
      xBound(2)=xv(j);
    end
    if yBound(1)>yv(j),
      yBound(1)=yv(j);
    end
    if yBound(2)<yv(j),
      yBound(2)=yv(j);
    end

    if isiBuket(posX,posY)==0,
      isiBuket(posX,posY)=isiBuket(posX,posY)+1;
      buket(posX,posY,isiBuket(posX,posY))=i;
    elseif buket(posX,posY,isiBuket(posX,posY))~=i,
      isiBuket(posX,posY)=isiBuket(posX,posY)+1;
      buket(posX,posY,isiBuket(posX,posY))=i;
    end
    %disp([xBound(1) xBound(2)]);
  end
  %if abs(xBound(2)-xBound(1))>1 || abs(yBound(2)-yBound(1))>1,
  % isi bucket yang berlebih ditolerir, namun bucket tidak boleh
  % kekurangan isi
    for k=xBound(1):xBound(2),
      for l=yBound(1):yBound(2),
        if inpolygon(k,l,xv,yv)==1,
          if isiBuket(k,l)==0,
            isiBuket(k,l)=isiBuket(k,l)+1;
            buket(k,l,isiBuket(k,l))=i;
            %disp('ok 1 : ');
            %disp([isiBuket(k,l) buket(posX,posY,isiBuket(posX,posY))]);
          elseif buket(k,l,isiBuket(k,l))~=i,
            isiBuket(k,l)=isiBuket(k,l)+1;
            buket(k,l,isiBuket(k,l))=i;
            %disp('ok 2 : ');
            %disp([isiBuket(k,l) buket(posX,posY,isiBuket(posX,posY))]);
          end
        end
      end
    end
    %end
end
%end buketing


% cek apakah bentuk yang bersangkutan adalah solid??
disp('apakah solid??');
%huu = isSolidModel(V,T,xMinMax,buket,isiBuket,lebarBucket,tinggiBucket,jmlBuketY);
kesolidan = isSolidModel2(V,T,xMinMax,buket,isiBuket,lebarBucket,jmlBuketY,jmlSegitiga);
disp(kesolidan);

% operasi inti
% menyimpan titik dan vektor normal setiap bagian yang dilewati pahat
vertexChecking = 0; % parameter adalah bagianXsurface,bagianYsurface,vertexCheckKe-,x,y,z
nVertexCheck = 0;  % parameter adalah bagianXsurface,bagianYsurface
vectorChecking = 0; % parameter adalah bagianXsurface,bagianYsurface,vertexCheckKe-,i,j,k
```

```
bagX = 0;  % index x untuk setiap bagian yang dilewati garis pahat
bagY = 0;  % index y untuk setiap bagian yang dilewati garis pahat
% pertanyaan :
% 1. bagaimana cara membedakan bagian bawah (untuk solid) dengan
% surfacenya, sebab keduanya masuk perhitungan
% 2. apakah yang diambil VN hanya bagian atas saja? Tidak, karena kalau
% bertumpuk vektor yang menghadap ke bawah tidak akan terdeteksi

% mencari ruang terbatas dan menampilkannya
doubleR = radiusTool*2;
indexTemp=0;   % banyaknya bagian yang bertumpuk
jarak = xMinMax(2)-xMinMax(1);
tinggi = 0;
pertengahanTinggi = 0;
counterSimpan = 0;
%disp([xMinMax(2)-xMinMax(1) radiusTool]);
%disp(jarak/radiusTool);
simpanTitikZDanVNormBertumpuk = 0; % z1-z2-z3-zn.., dan menyimpan vektor normal nya
jumlahTitikZBertumpuk = 0; % menyimpan jumlah titik z yang bertumpuk
vertexOperation = 0;   % menyimpan vertex-vertex yang diperiksa, argumen terdiri dari index x,index y,
x/y/z/i/j/k, 0 semua jika bertumpuk
jmlTitikPadaBaris = 0; % menyimpan banyaknya vertex pada baris tertentu
x = xMinMax(1)-radiusTool;
%for x=xMinMax(1)+radiusTool:doubleR:xMinMax(2),
while 1,
   x = x+doubleR;
   if x>=xMinMax(2),
      break;
   end
   % untuk bagian surface yang dilewati garis pahat
   bagX = bagX+1;
   bagY = 0;
   % end of untuk bagian surface yang dilewati garis pahat
   indexSimpan = 0;   % array menyimpan segitiga
   cekPertama = 0;
   for y=yMinMax(1)+radiusTool:doubleR:yMinMax(2),
      % untuk bagian surface yang dilewati garis pahat
      bagY = bagY+1;
      nVertexCheck(bagX,bagY) = 0;   % pada bagian ke (bagX,bagY)
      % end of untuk bagian surface yang dilewati garis pahat
      posX = floor(x/lebarBucket)+1;
      posY = floor(y/tinggiBucket)+1;
      Vpotong.x=x;
      Vpotong.y=y;
      jmlSgtBerpot = 0;  % lakukan sesuatu jika ada segitiga yang bertumpuk
      banyak = isiBuket(posX,posY);
      for i=1:banyak,
         indexTri = buket(posX,posY,i);
         dd=0;
         aa=3;
         if indexTri>0,
            for j=1:3,
               index1 = j;
               index2 = j+1;
               if index2>3,
                  index2=index2-3;
```

```
            end

            % vertex pertama garis
            Vertex1.x = V(T(indexTri,index1),1);
            Vertex1.y = V(T(indexTri,index1),2);
            % vertex kedua garis
            Vertex2.x = V(T(indexTri,index2),1);
            Vertex2.y = V(T(indexTri,index2),2);
            %hitung
            t=-1*(Vertex2.y-Vertex1.y)*Vpotong.x+(Vertex2.x-
Vertex1.x)*Vpotong.y+Vertex1.x*Vertex2.y-Vertex2.x*Vertex1.y;
            %disp([t i]);
            if t>0,
               dd=dd+1;
            elseif t<0,
               aa=aa-1;
            else
               aa=aa-1;
               dd=dd+1;
            end
         end
      end
      if dd==3 || aa==0,
         jmlSgtBerpot=jmlSgtBerpot+1;
         %disp(jmlSgtBerpot);
         indexSimpan(jmlSgtBerpot)=indexTri;

         % untuk bagian surface yang dilewati garis pahat
         % cari titik tengah segitiga
         D = [x y];
         Ver1 = [V(T(indexTri,1),1) V(T(indexTri,1),2) V(T(indexTri,1),3)];
         Ver2 = [V(T(indexTri,2),1) V(T(indexTri,2),2) V(T(indexTri,2),3)];
         Ver3 = [V(T(indexTri,3),1) V(T(indexTri,3),2) V(T(indexTri,3),3)];
         hasilZ = hitung_z2(Ver1,Ver2,Ver3,D);

         nVertexCheck(bagX,bagY) = nVertexCheck(bagX,bagY)+1;
         % simpan xyz titik potong
         vertexChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = Vpotong.x;   % x
         vertexChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = Vpotong.y;   % y
         vertexChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = hasilZ(1);   % z
         % simpan ijk titik potong
         if hasilZ(2) == 0,
            % titik potong jatuh pada vertex segitiga
            if hasilZ(3) == 1,
               % titik potong jatuh pada vertex pertama segitiga
               vektorNormal1 =
hitungVekNorBerbobot2(V,T,buket,isiBuket,T(indexTri,1),lebarBucket,tinggiBucket);
            elseif hasilZ(3) == 2,
               % titik potong jatuh pada vertex kedua segitiga
               vektorNormal1 =
hitungVekNorBerbobot2(V,T,buket,isiBuket,T(indexTri,2),lebarBucket,tinggiBucket);
            elseif hasilZ(3) == 3,
               % titik potong jatuh pada vertex ketiga segitiga
               vektorNormal1 =
hitungVekNorBerbobot2(V,T,buket,isiBuket,T(indexTri,3),lebarBucket,tinggiBucket);
            end
```

```
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = vektorNormal1(1);   % i
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = vektorNormal1(2);   % j
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = vektorNormal1(3);   % k
         elseif hasilZ(2) == 1,
            % titik potong jatuh pada edge segitiga
            if hasilZ(3) == 1,
               % titik potong jatuh pada edge segitiga dari vertex
               % pertama dan kedua
               index1 = T(indexTri,1);
               index2 = T(indexTri,2);
            elseif hasilZ(3) == 2,
               % titik potong jatuh pada edge segitiga dari vertex
               % kedua dan ketiga
               index1 = T(indexTri,2);
               index2 = T(indexTri,3);
            elseif hasilZ(3) == 3,
               % titik potong jatuh pada vertex ketiga segitiga
               % titik potong jatuh pada edge segitiga dari vertex
               % ketiga dan pertama
               index1 = T(indexTri,3);
               index2 = T(indexTri,1);
            end
            % hitung vektor normal masing-masing vertex
            vektorNormal1 =
hitungVekNorBerbobot2(V,T,buket,isiBuket,index1,lebarBucket,tinggiBucket);
            vektorNormal2 =
hitungVekNorBerbobot2(V,T,buket,isiBuket,index2,lebarBucket,tinggiBucket);
            % masukkan nilai xyz ke vertex1 dan vertex2
            vertex1(1) = V(index1,1);
            vertex1(2) = V(index1,2);
            vertex1(3) = V(index1,3);
            vertex2(1) = V(index2,1);
            vertex2(2) = V(index2,2);
            vertex2(3) = V(index2,3);
            vertexTengah(1) = Vpotong.x;
            vertexTengah(2) = Vpotong.y;
            vertexTengah(3) = hasilZ(1);
            % hitung normal vertexnya
            vektorNormal =
hitungVekNorInterNormVer(vektorNormal1,vektorNormal2,vertex1,vertex2,vertexTengah);
            % masukkan ke dalam vectorChecking
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = vektorNormal(1);   % i
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = vektorNormal(2);   % j
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = vektorNormal(3);   % k
         else
            % titik potong jatuh pada face segitiga
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),1) = T(indexTri,4);   % i
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),2) = T(indexTri,5);   % j
            vectorChecking(bagX,bagY,nVertexCheck(bagX,bagY),3) = T(indexTri,6);   % k
         end
         % untuk bagian surface yang dilewati garis pahat
      end
   end
   % lihat apakah solid type lebih dari 1 atau tidak
   n = 1;
   % hilangkan yang boundary tertutup
```

```
if jmlSgtBerpot==2 && kesolidan==1,
   % bagian yang diambil adalah atas atau bawah?
   if vertexChecking(bagX,bagY,1,3)<vertexChecking(bagX,bagY,2,3),
      n = 2;
   end
   vertexOperation(bagX,bagY,1) = vertexChecking(bagX,bagY,n,1); % x
   vertexOperation(bagX,bagY,2) = vertexChecking(bagX,bagY,n,2); % y
   vertexOperation(bagX,bagY,3) = vertexChecking(bagX,bagY,n,3); % z
   vertexOperation(bagX,bagY,4) = vectorChecking(bagX,bagY,n,1); % i
   vertexOperation(bagX,bagY,5) = vectorChecking(bagX,bagY,n,2); % j
   vertexOperation(bagX,bagY,6) = vectorChecking(bagX,bagY,n,3); % k
   continue;
elseif jmlSgtBerpot == 1,
   % hanya 1 layer
   vertexOperation(bagX,bagY,1) = vertexChecking(bagX,bagY,n,1); % x
   vertexOperation(bagX,bagY,2) = vertexChecking(bagX,bagY,n,2); % y
   vertexOperation(bagX,bagY,3) = vertexChecking(bagX,bagY,n,3); % z
   vertexOperation(bagX,bagY,4) = vectorChecking(bagX,bagY,n,1); % i
   vertexOperation(bagX,bagY,5) = vectorChecking(bagX,bagY,n,2); % j
   vertexOperation(bagX,bagY,6) = vectorChecking(bagX,bagY,n,3); % k
   continue;
elseif jmlSgtBerpot > 0,
   sejajar = 1;
   tinggiZ = vertexChecking(bagX,bagY,1,3); % z
   for i=2:nVertexCheck(bagX,bagY),
      if tinggiZ~=vertexChecking(bagX,bagY,i,3),
         sejajar=0;
         break;
      end
   end
   if sejajar==1,
      vertexOperation(bagX,bagY,1) = vertexChecking(bagX,bagY,n,1); % x
      vertexOperation(bagX,bagY,2) = vertexChecking(bagX,bagY,n,2); % y
      vertexOperation(bagX,bagY,3) = vertexChecking(bagX,bagY,n,3); % z
      vertexOperation(bagX,bagY,4) = vectorChecking(bagX,bagY,n,1); % i
      vertexOperation(bagX,bagY,5) = vectorChecking(bagX,bagY,n,2); % j
      vertexOperation(bagX,bagY,6) = vectorChecking(bagX,bagY,n,3); % k
      continue;
   end
end
% jika terdapat segitiga yang bertumpuk, cari tingginya
if jmlSgtBerpot>1,

   % cek apakah perpotongan tersebut berada pada titik yang
   % berbeda (jika berpotongan pada garis / titik)

   % urutkan letak titik perpotongan dari titik terkecil ke titik
   % terbesar. Jangan lupa pastikan titik yang sama persis harus
   % dihapus

   % buat garis berselang-seling antara titik terkecil dengan
   % titik terbesar

   indexTemp=indexTemp+1;
   % simpan titik yang berpotongan
   %D = [x y];
```

```
huhu=0;
arrayTemp = 0; % untuk menyimpan array Z temporary, for pengurutan only
aa = 1;
for i=1:jmlSgtBerpot,
   % cari titik tengah segitiga
   %Ver1 = [V(T(indexSimpan(i),1),1) V(T(indexSimpan(i),1),2) V(T(indexSimpan(i),1),3)];
   %Ver2 = [V(T(indexSimpan(i),2),1) V(T(indexSimpan(i),2),2) V(T(indexSimpan(i),2),3)];
   %Ver3 = [V(T(indexSimpan(i),3),1) V(T(indexSimpan(i),3),2) V(T(indexSimpan(i),3),3)];
   %D(3) = hitung_z(Ver1,Ver2,Ver3,D);
   % simpan titik tersebut
   ashoyy(indexTemp,i,1) = vertexChecking(bagX,bagY,i,1);
   ashoyy(indexTemp,i,2) = vertexChecking(bagX,bagY,i,2);
   ashoyy(indexTemp,i,3) = vertexChecking(bagX,bagY,i,3);

   %ashoyy(indexTemp,i,1) = D(1);  % ashoyy(urutan tiang, vertex atas/bawah, elemen titik x/y/z)
   %ashoyy(indexTemp,i,2) = D(2);
   %ashoyy(indexTemp,i,3) = D(3);
   % tambahkan z untuk menghitung pertengahan
   huhu=huhu+ashoyy(indexTemp,i,3);

   % untuk menghitung segitiga bertumpuk yang lebih dari 1
   %arrayTemp(i) = D(3);
   arrayTemp(aa) = vertexChecking(bagX,bagY,i,3);
   aa=aa+1;
   arrayTemp(aa) = vectorChecking(bagX,bagY,i,1);
   aa=aa+1;
   arrayTemp(aa) = vectorChecking(bagX,bagY,i,2);
   aa=aa+1;
   arrayTemp(aa) = vectorChecking(bagX,bagY,i,3);
   aa=aa+1;
end
tambahan=4;
% urutkan z bertumpuk dari yang terendah hingga yang tertinggi
[m n] = size(arrayTemp);
arrayTemp = lakukanInsertionsort2(arrayTemp,1,n,1,tambahan);
% untuk menghitung segitiga bertumpuk yang lebih dari 1
% masukkan segitiga bertumpuk tersebut ke tempat yang telah
% disediakan
simpanTitikZDanVNormBertumpuk(indexTemp,1,1) = arrayTemp(1);
simpanTitikZDanVNormBertumpuk(indexTemp,1,2) = arrayTemp(2);
simpanTitikZDanVNormBertumpuk(indexTemp,1,3) = arrayTemp(3);
simpanTitikZDanVNormBertumpuk(indexTemp,1,4) = arrayTemp(4);
counter = 1;  % membantu mengecek penumpukan
akhir = jmlSgtBerpot*tambahan;
i = 1+tambahan;
%for i=1+tambahan:tambahan:akhir,
while i<=akhir,
   if abs(arrayTemp(i)-arrayTemp(i-tambahan))==0.0001,  % memastikan bahwa titik yang
bersangkutan benar-benar bertumpuk
      % do nothing
      %continue;
   else
      counter=counter+1;
      simpanTitikZDanVNormBertumpuk(indexTemp,counter,1) = arrayTemp(i);
      simpanTitikZDanVNormBertumpuk(indexTemp,counter,2) = arrayTemp(i+1);
      simpanTitikZDanVNormBertumpuk(indexTemp,counter,3) = arrayTemp(i+2);
```

```
            simpanTitikZDanVNormBertumpuk(indexTemp,counter,4) = arrayTemp(i+3);
          end
          i = i+tambahan;
        end
        jumlahTitikZBertumpuk(indexTemp) = counter;
        if counter>1,
          % set 0 semua
          vertexOperation(bagX,bagY,1) = 0;
          vertexOperation(bagX,bagY,2) = 0;
          vertexOperation(bagX,bagY,3) = 0;
          vertexOperation(bagX,bagY,4) = 0;
          vertexOperation(bagX,bagY,5) = 0;
          vertexOperation(bagX,bagY,6) = 0;
        end
      end
    end
    jmlTitikPadaBaris(bagX)=bagY;
end
fclose(fid);

% strategi :
% 1. cek dengan cara yang sudah ada
% 2. cek kesadelan dengan mengecek kiri dan kanan dari titik periksa
% 3. cek perbedaan ketinggian antar titik periksa

% cek searah sumbu y
% kode vektor normal
% 0 : bertumpuk
% 1 : miring ke depan
% 2 : datar
% 3 : miring ke belakang
areaCheck = 0;
vektor = [0 0 0];
for i=1:bagX,
    for j=1:jmlTitikPadaBaris(i),
        areaCheck(i,j) = 2;  % nilai default
        if vertexOperation(i,j,4)~=0 || vertexOperation(i,j,5)~=0 || vertexOperation(i,j,6)~=0,
            % cegah bukan pada bagian bertumpuk
            vektor(1) = vertexOperation(i,j,4);
            vektor(2) = vertexOperation(i,j,5);
            vektor(3) = vertexOperation(i,j,6);
            if isMoreThanNDegree(vektor,kemiringan)==1,
                if vektor(2)>0, % cek y-nya
                    areaCheck(i,j) = 1;
                elseif vektor(2)<0,
                    areaCheck(i,j) = 3;
                end
            end
        else
            % bertumpuk
            areaCheck(i,j) = 0;
        end
    end
end

% jika 0-2-3 / 0-3 : identifikasi
```

```
% jika 1-2-0 / 1-0 : identifikasi
% jika 1-2-3 / 1-3 : identifikasi
% jika 2-2-2-.... : cek tinggi diantaranya
% mulai proses : 1 / 0
% stop proses : 3
% jika perbedaan tinggi > lebar pahat : identifikasi
areaCheck2 = 0;
for i=1:bagX,
   inProses=0;
   lastVekCheck = 2;  % nilai default
   currentVekCheck = 2;
   % jika awalan bernilai 1 (cekungan pertama, proses sedang in)
   lastVekCheck = areaCheck(i,1);
   if lastVekCheck == 1,
      inProses=1;
      disp('pertama sudah ok');
      disp(i);
   end
   simpan(1)=0;
   simpan(2)=-1;
   for j=2:jmlTitikPadaBaris(i),
      areaCheck2(i,j) = 0;  % nilai default
      isSadel(i,j) = 0; % nilai default
      kiri = 0;
      kanan=0;
      currentVekCheck = areaCheck(i,j);
      if inProses==0,
         if currentVekCheck==1 && lastVekCheck==2,
            inProses=1;
         elseif currentVekCheck==1 && lastVekCheck==3,
            inProses=1;
         elseif currentVekCheck==2 && lastVekCheck==0,
            inProses=1;
            simpan(1)=1;   % menandakan keluar dari CBV, dan bertemu datar
            simpan(2)=j;   % menyimpan letak keluar dari CBV
         elseif currentVekCheck==3 && lastVekCheck==0,
            inProses=1;
         end
      else
         if currentVekCheck==2 && lastVekCheck==3,
            inProses=0;
         elseif currentVekCheck==1 && lastVekCheck==2,  % jika baru keluar dari CBV, dan bertemu
datar, kemudian ketemu cekung
            inProses=1;
            if simpan(1)==1,
               % nolkan sebelumnya
               for k=j:-1:simpan(2),
                  areaCheck2(i,k) = 0;  % nilai default
               end
               simpan(1) = 0;  % reset nilainya
               simpan(2) = -1; % reset nilainya
            end
         elseif currentVekCheck==1 && lastVekCheck==3,
            % berhenti dan mulai lagi secara bersamaan
            inProses=0;
            inProses=1;
```

```
        elseif currentVekCheck==0 && lastVekCheck==1,
          inProses=0;
        elseif currentVekCheck==0 && lastVekCheck==2,
          inProses=0;
        end
        % sebelum memberi nilai, periksa dulu kesadelan titik yang
        % bersangkutan
        if i>1,
          % cek sebelah kiri
          kiri = cekSamping(i-1,jmlTitikPadaBaris,vertexOperation,2,vertexOperation(i,j,2));
        end
        if i<bagX,
          % cek sebelah kanan
          kanan = cekSamping(i+1,jmlTitikPadaBaris,vertexOperation,2,vertexOperation(i,j,2));
        end

        if kiri==-1 && kanan==1,
          % keadaan sadel
          isSadel(i,j) = 1;
        else
          % tidak sadel
          isSadel(i,j) = 0;
        end
        areaCheck2(i,j)=1;
      end
      lastVekCheck = areaCheck(i,j);
    end
    % jika terakhirnya 1, dan bukan berada pada code 3 (cekungan terakhir),
    % maka harus mundur, dan nol-kan hingga bertemu 0, atau hingga bertemu
    % perubahan code 1-2/1-3 dari belakang
    j = jmlTitikPadaBaris(i);
    if areaCheck(i,j)~=3,  % jika terakhir bukan code 3

%       disp('ada mundur...');
%       disp(j);
      currentVekCheck = areaCheck(i,j);
      lastVekCheck = areaCheck(i,j);
      while areaCheck2(i,j)==1,  % selama masih dianggap boundary volume
        areaCheck2(i,j)=0; % nolkan
        currentVekCheck = areaCheck(i,j);
        if j==1,  % jika sudah titik awal, keluar
          break;
        elseif currentVekCheck==2 && lastVekCheck==1,  % jika ada perubahan 1-2, keluar
          break;
        elseif currentVekCheck==3 && lastVekCheck==1,  % jika ada perubahan 1-3, keluar
          break;
        end
        lastVekCheck = areaCheck(i,j);
        j = j-1;
      end
    end
end
% end of cek searah sumbu y

% mengecek berdasarkan ketinggian
% tinggi diperiksa berdasarkan letak antar 2 titik periksa
```

```
% kode 0 = sama tinggi
% kode 1 = tinggi ke rendah
% kode 2 = rendah ke tinggi
% kode -1 = daerah bertumpuk, tidak dicek
cekTinggi=0;
heightCheck=0;
for i=1:bagX,
  heightCheck(i,1)=0;
  for j=2:jmlTitikPadaBaris(i),
    heightCheck(i,j)=0; % default
    if vertexOperation(i,j-1,4)==0 && vertexOperation(i,j-1,5)==0 && vertexOperation(i,j-1,6)==0,
      % daerah bertumpuk
      heightCheck(i,j)=-1;
    elseif vertexOperation(i,j-1,3)-heightTolerance>vertexOperation(i,j,3),
      % turun
      heightCheck(i,j)=1;
    elseif vertexOperation(i,j-1,3)+heightTolerance<vertexOperation(i,j,3),
      % naik
      heightCheck(i,j)=2;
    else
      % tingginya sama
      heightCheck(i,j)=0;
    end
  end
end
% jika 1-0-2 / 1-2 : identifikasi
heightCheck2=0;
for i=1:bagX,
  inProses=0;
  for j=1:jmlTitikPadaBaris(i),
    heightCheck2(i,j)=0;   % default
    if inProses==0,
      if heightCheck(i,j)==1,
        inProses=1;
      end
    else
      heightCheck2(i,j)=1;
      if heightCheck(i,j)==-1,
        % daerah bertumpuk
        heightCheck2(i,j)=0;
        inProses=0;
      end
      if heightCheck(i,j)==2, % jika rendah ke tinggi, lihat langkah selanjutnya
        if j<jmlTitikPadaBaris(i),
          if heightCheck(i,j+1)==0 || heightCheck(i,j+1)==1,  % jika selanjutnya datar atau turun
            heightCheck2(i,j)=0;
            inProses=0;
          end
        end
      end
    end
  end
  % jika terakhir bukan pada code 2(naik), mundur
  j = jmlTitikPadaBaris(i);
  if heightCheck(i,j)~=2,
    while heightCheck2(i,j)==1, % selama masih dianggap boundary volume
```

```
        if j==0,   % sudah berada di titik awal
          break;
        elseif heightCheck(i,j)==2, % jika sudah berada pada posisi naik, berhenti
          break;
        end
        heightCheck2(i,j)=0;
        j=j-1;
      end
    end
end
% end of mengecek berdasarkan ketinggian

% tampilkan setiap vertex dan vektor normalnya, kec yang bertumpuk
for i=1:bagX,
  for j=1:jmlTitikPadaBaris(i),
    if (areaCheck2(i,j)==1 || heightCheck2(i,j)==1) && isSadel(i,j)==0,
      if (vertexOperation(i,j,4)~=0 || vertexOperation(i,j,5)~=0 || vertexOperation(i,j,6)~=0) &&
vertexOperation(i,j,6)>0,
        % titik pertama
%         A(1) = vertexOperation(i,j,1)+deltaX;
%         B(1) = vertexOperation(i,j,2)+deltaY;
%         C(1) = vertexOperation(i,j,3)+deltaZ;
        % titik kedua
%         A(2) = vertexOperation(i,j,1) + deltaX + vertexOperation(i,j,4)*10;  % x+i
%         B(2) = vertexOperation(i,j,2) + deltaY + vertexOperation(i,j,5)*10;  % y+j
%         C(2) = vertexOperation(i,j,3) + deltaZ + vertexOperation(i,j,6)*10;  % z+k

        A(1) = vertexOperation(i,j,1);
        B(1) = vertexOperation(i,j,2);
        C(1) = vertexOperation(i,j,3);
        % titik kedua
        A(2) = vertexOperation(i,j,1) + vertexOperation(i,j,4)*10;  % x+i
        B(2) = vertexOperation(i,j,2) + vertexOperation(i,j,5)*10;  % y+j
        C(2) = vertexOperation(i,j,3) + vertexOperation(i,j,6)*10;  % z+k
        vertexOperation(i,j,6)*10;  % z+k
        % gambarkan vektor normal
        plot3(A,B,C,'k');
        hold on;
      end
    end
  end
end

posisiAwalBottomTool = [0 0 0];    % untuk menggambar pahat posisi awal
posisiAwalTopTool = [0 0 posisiAwalBottomTool+panjangTool];
if indexTemp>0,
  % persiapan penggambaran garis tengah
  patokan = 0;       % sebagai patokan y dan z, array 2d dengan el 1 adalah... dan el2 adalah yz
  jmlPatokan = 0;    % jumlah patokan y dan z
  garisMelintang = 0; % garis melintang yang akan digambar, array 2d dengan el1 adalah ... dan el2 adalah
xyz
  jmlTtkGaris = 0;   % jml titik pada satu garis melintang
  % end of persiapan penggambaran garis tengah

  % persiapan penggambaran garis tengah kedua
  patokan2 = 0;       % sebagai patokan y dan z, array 2d dengan el 1 adalah... dan el2 adalah yz
```

```
  jmlPatokan2 = 0;    % jumlah patokan y dan z
  garisMembujur = 0; % garis melintang yang akan digambar, array 2d dengan el1 adalah ... dan el2 adalah
xyz
  jmlTtkGaris2 = 0;   % jml titik pada satu garis melintang
  % end of persiapan penggambaran garis tengah

  % tampilkan garis tinggi yang kedua
  for i=1:indexTemp,
    awalan=1;
    if kesolidan==1,
      if mod(jumlahTitikZBertumpuk(i),2)==1,
        awalan=1;
      else
        awalan=2;
      end
    end
    for j=awalan:2:jumlahTitikZBertumpuk(i)-1,
      % titik pertama
      A(1) = ashoyy(i,1,1);
      B(1) = ashoyy(i,1,2);
      C(1) = simpanTitikZDanVNormBertumpuk(i,j,1);
      % titik kedua
      A(2) = ashoyy(i,2,1);
      B(2) = ashoyy(i,2,2);
      C(2) = simpanTitikZDanVNormBertumpuk(i,j+1,1);
      % gambarkan
      plot3(A,B,C,'y');
      hold on;
      % gambarkan titik tengah
      tengah = (C(1)+C(2))/2;
      plot3(A(1),B(1),tengah,'g+');
      hold on;

      % analisis patokan pertama
      tempPat = searchIndexBy2Index(jmlPatokan,patokan,B(1),tengah,0.1,zTolerance);
      if tempPat>jmlPatokan,
        jmlPatokan=jmlPatokan+1;
        jmlTtkGaris(tempPat) = 0;
      end
      patokan(tempPat,1) = B(1);
      patokan(tempPat,2) = tengah;
      % simpan garis melintang
      jmlTtkGaris(tempPat) = jmlTtkGaris(tempPat)+1;
      garisMelintang(tempPat,jmlTtkGaris(tempPat),1) = A(1);
      garisMelintang(tempPat,jmlTtkGaris(tempPat),2) = B(1);
      garisMelintang(tempPat,jmlTtkGaris(tempPat),3) = tengah;

      % analisis patokan kedua
      tempPat = searchIndexBy2Index(jmlPatokan2,patokan2,A(1),tengah,0.1,zTolerance);
      if tempPat>jmlPatokan2,
        jmlPatokan2=jmlPatokan2+1;
        jmlTtkGaris2(tempPat) = 0;
      end
      patokan2(tempPat,1)=A(1);
      patokan2(tempPat,2)=tengah;
      %simpan garis membujur
```

```
      jmlTtkGaris2(tempPat) = jmlTtkGaris2(tempPat)+1;
      garisMembujur(tempPat,jmlTtkGaris2(tempPat),1)=A(1);
      garisMembujur(tempPat,jmlTtkGaris2(tempPat),2)=B(1);
      garisMembujur(tempPat,jmlTtkGaris2(tempPat),3)=tengah;
   end
end
% gambarkan garis melintang
for i=1:jmlPatokan,
   A=0; B=0; C=0;
   for j=1:jmlTtkGaris(i),
      A(j) = garisMelintang(i,j,1);
      B(j) = garisMelintang(i,j,2);
      C(j) = garisMelintang(i,j,3);
   end
   plot3(A,B,C,'r');
   hold on;
end
% gambarkan garis membujur sekaligus vektornya
for i=1:jmlPatokan2,
   A=0; B=0; C=0;
   vektorGaris = [0 0 0];
   for j=1:jmlTtkGaris2(i),
      A(j) = garisMembujur(i,j,1);   % sudah termasuk nilai delta x,y,z
      B(j) = garisMembujur(i,j,2);
      C(j) = garisMembujur(i,j,3);
      if j>1,
         vektorGaris(1) = vektorGaris(1) + (A(j)-A(j-1));
         vektorGaris(2) = vektorGaris(2) + (B(j)-B(j-1));
         vektorGaris(3) = vektorGaris(3) + (C(j)-C(j-1));
      end
   end
   plot3(A,B,C,'c');
   hold on;
   % gambar vektor pahat
   if vektorGaris(1)~=0 || vektorGaris(2)~=0 || vektorGaris(3)~=0,
      vektorGaris = makeVektorUnit(vektorGaris);

      A=0; B=0; C=0;
      % menggambar tool pada saat berada di close boundary volume
      % titik pangkal pahat berada pada boundary volume terluar
      A(1) = garisMembujur(i,jmlTtkGaris2(i),1);
      B(1) = garisMembujur(i,jmlTtkGaris2(i),2);
      C(1) = garisMembujur(i,jmlTtkGaris2(i),3);

      A(2) = garisMembujur(i,jmlTtkGaris2(i),1)+vektorGaris(1)*panjangTool;
      B(2) = garisMembujur(i,jmlTtkGaris2(i),2)+vektorGaris(2)*panjangTool;
      C(2) = garisMembujur(i,jmlTtkGaris2(i),3)+vektorGaris(3)*panjangTool;


      % menggambar tool pada posisi awal
      % yang sama pada satu posisi adalah x-nya, sedangkan y dan z nya
      % berbeda
      % posisi awal pahat adalah setinggi z max ditambah dengan
      % tinggi pahat
      mulaiA=0; mulaiB=0; mulaiC=0;
      mulaiA(1) = garisMembujur(i,jmlTtkGaris2(i),1);
```

```
        mulaiB(1) = yMinMax(2);
        mulaiC(1) = zMinMax(2);
        mulaiA(2) = garisMembujur(i,jmlTtkGaris2(i),1);
        mulaiB(2) = yMinMax(2);
        mulaiC(2) = zMinMax(2)+panjangTool;
        plot3(mulaiA,mulaiB,mulaiC,'m');
        hold on;

        % menggambar proses perjalanan pahat
        garisBawahTool = [A(1)-mulaiA(1) B(1)-mulaiB(1) C(1)-mulaiC(1)];
        garisBawahTool = determineVektorLength(garisBawahTool,banyakAnimasi);
        garisAtasTool = [A(2)-mulaiA(2) B(2)-mulaiB(2) C(2)-mulaiC(2)];
        garisAtasTool = determineVektorLength(garisAtasTool,banyakAnimasi);

        % init
        Ajalan(1) = mulaiA(1);  % bawah tool
        Bjalan(1) = mulaiB(1);
        Cjalan(1) = mulaiC(1);

        Ajalan(2) = mulaiA(2);  % atas tool
        Bjalan(2) = mulaiB(2);
        Cjalan(2) = mulaiC(2);

        for j=1:banyakAnimasi,
          Ajalan(1) = Ajalan(1)+garisBawahTool(1);  % bawah tool
          Bjalan(1) = Bjalan(1)+garisBawahTool(2);
          Cjalan(1) = Cjalan(1)+garisBawahTool(3);

          Ajalan(2) = Ajalan(2)+garisAtasTool(1);  % atas tool
          Bjalan(2) = Bjalan(2)+garisAtasTool(2);
          Cjalan(2) = Cjalan(2)+garisAtasTool(3);

          plot3(Ajalan,Bjalan,Cjalan,'m');
          hold on;
        end
      end
    end
end

% untuk penggambaran
for i=1:jmlSegitiga,
    iv1 = T(i,1);
    iv2 = T(i,2);
    iv3 = T(i,3);

    A = [V(iv1,1) V(iv2,1) V(iv3,1)];
    B = [V(iv1,2) V(iv2,2) V(iv3,2)];
    C = [V(iv1,3) V(iv2,3) V(iv3,3)];

    % untuk penggambaran
    plot3(A,B,C);
    hold on;
end

%disp([xMinMax(1) xMinMax(2) yMinMax(1) yMinMax(2) zMinMax(1) zMinMax(2)]);
```

```
%axis([xMinMax(1)+deltaX xMinMax(2)+deltaX yMinMax(1)+deltaY yMinMax(2)+deltaY
zMinMax(1)+deltaZ zMinMax(2)+deltaZ]);

axis([xMinMax(1) xMinMax(2) yMinMax(1) yMinMax(2) zMinMax(1) zMinMax(2)+panjangTool]); %
gambarkan x,y,z sesudah terpengaruh delta, bukan seperti dahulu
```