Path Planning and Obstacle Avoidance of Unmanned Aerial Vehicles Using 3D Shapes and Time elastic Bands

This paper is in reference to "Elastic Bands: Connecting Path Planning and Control - Sean Quinlan and Oussama Khatib."

In the referenced paper, the concept of bubbles and their overlap has been used to propose a novel method of path planning in 2D. At the time of publishing, this was considered a breakthrough and is still a highly cited piece of work. I propose that this concept be extended to 3-dimensional bubbles, and their overlap be used as a factor for path generation and obstacle avoidance.

Obstacles will be covered with three-dimensional spheres, which will help navigate the path using an elastic band. The ratios of the sphere can be varied using three-dimensional space geometry, which will give us the freedom to use ellipses in case of obstacles of uneven shape. Moreover, for obstacles such as poles and towers can be considered as cylinders, hence helping us distinguish between and characterize obstacles. This simple concept will help us characterize the flyability around different types of obstacles. For example, if an obstacle is covered by a sphere, this gives us the information that we have an infinite number of ways to cross the obstacle. We can follow any available surface curvature around the sphere, and we would have successfully avoided the obstacle. Similarly, if an obstacle is covered by an infinite vertical cylinder (for example, a very high tower), this gives us the information that we can follow the curved surface of the sphere to avoid it but cannot go above or below the obstacle. This analogy can be used for any kind of regular or irregular shape. Once we have assigned the three-dimensional object a three-dimensional shape, the computation for a path becomes very simple because we just have to follow the surface lines of the defined object to get from one side to the other.

Here we can use the concept of Time Elastic Bands or simply Elastic Bands. Initially, non-forced elastic bands stretching from our initial start to our goal. As we assign shapes in real-time, the elastic band will experience a repulsive force which will ensure that it deforms due to obstacles in its path. Moreover, a constrain will have to be implemented to ensure that the drone follows the shortest and smoothest curvature along curved bodies. This will ensure that we generate the shortest path without using any optimizing algorithms on a global scale.

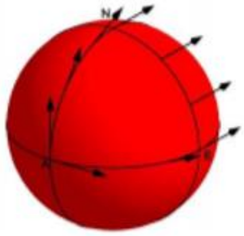


Figure 1.0: Infinite Methods to Get Past a Sphere

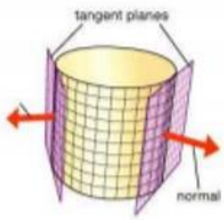
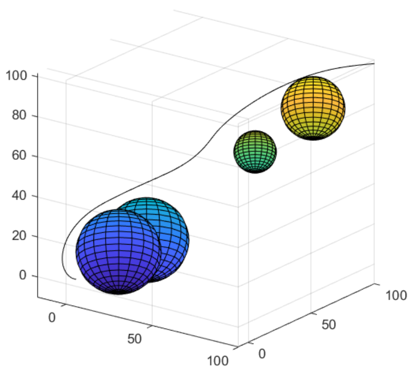
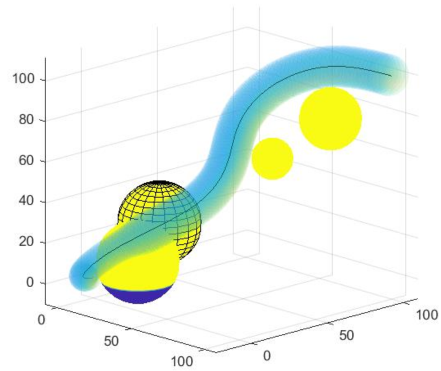


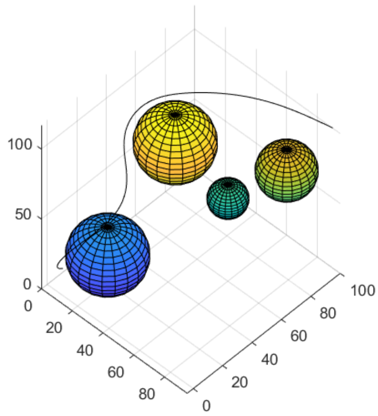
Figure 1.1: Methods to Get Past a Cylinder Along Its Curved Surface

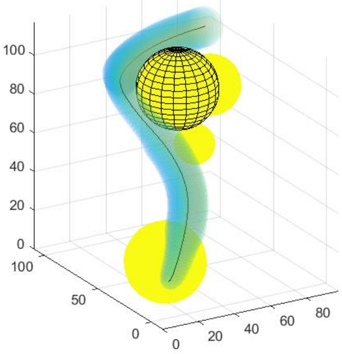
MATLAB SIMULATIONS IN 3D

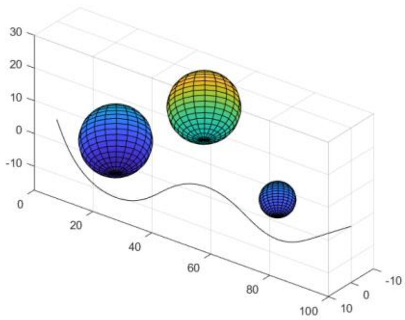
1) OBSTACLES AS SPHERES

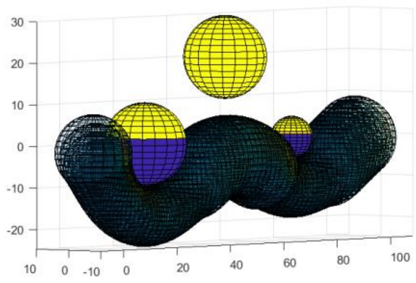


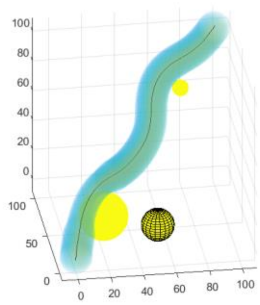


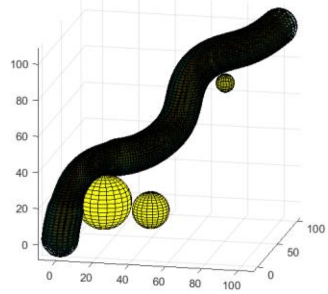




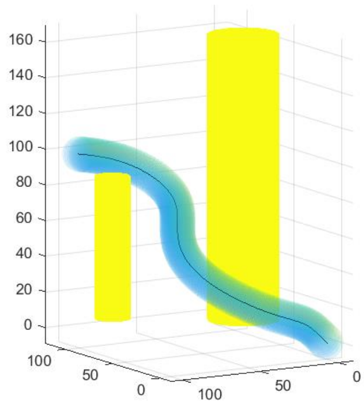


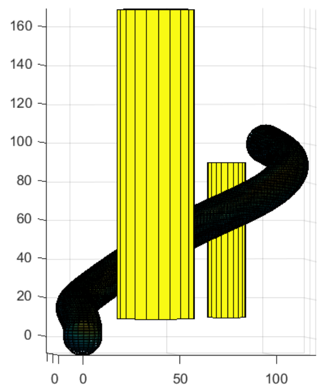


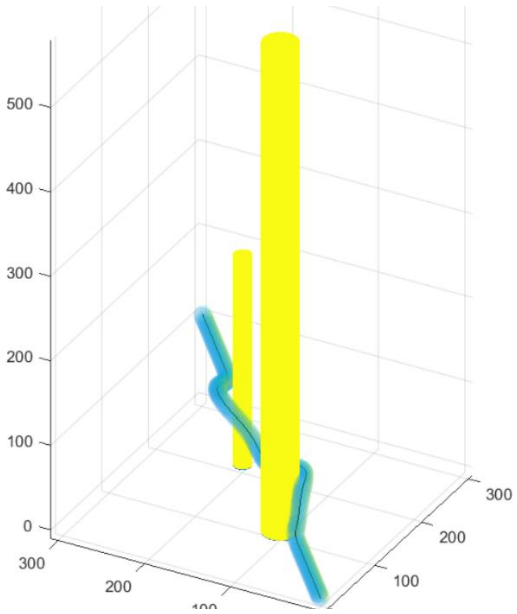


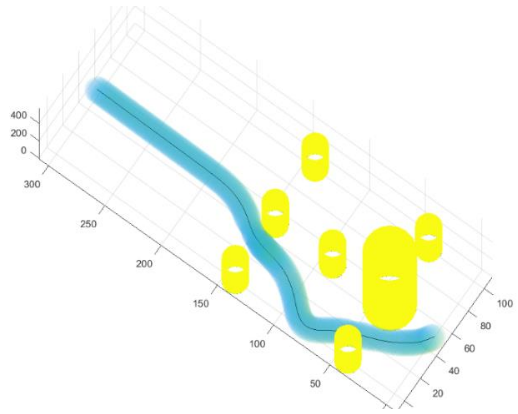


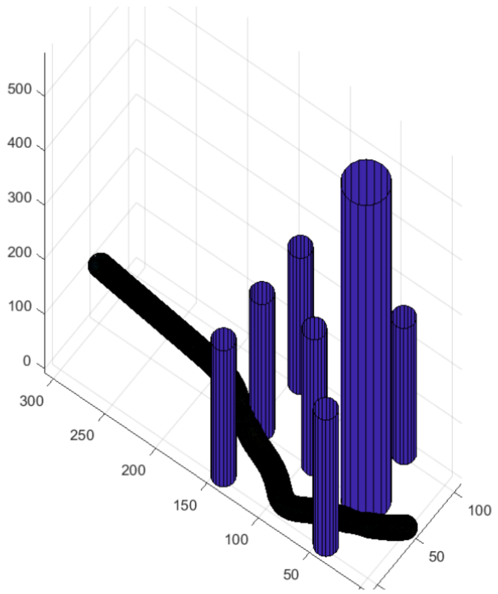
2) OBSTACLES AS CYLINDERS

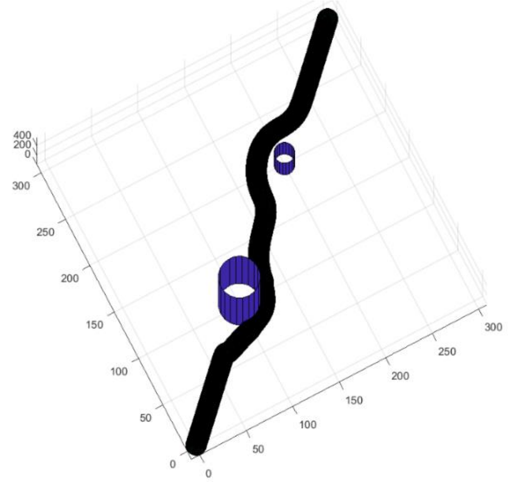




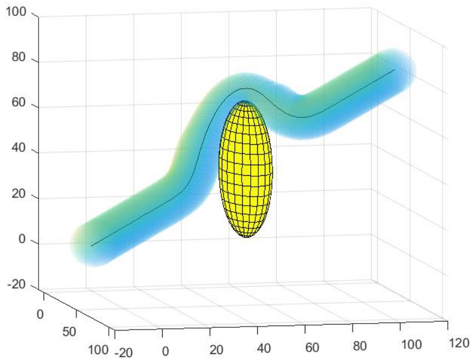


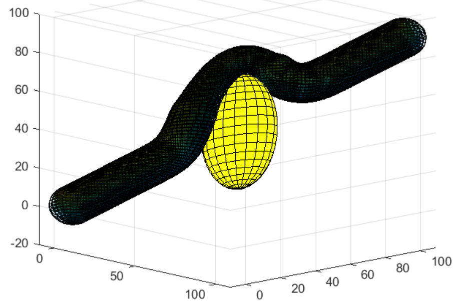


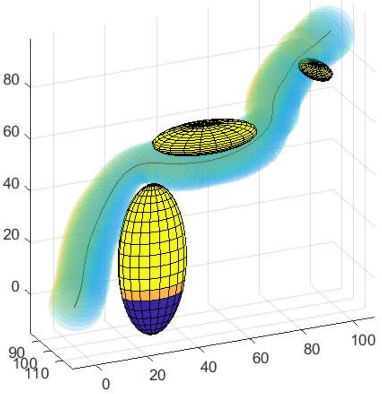


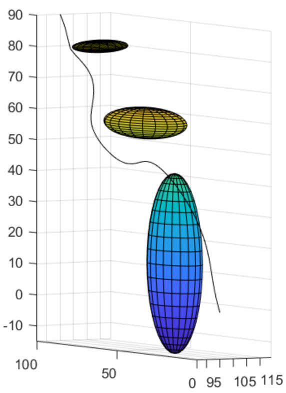


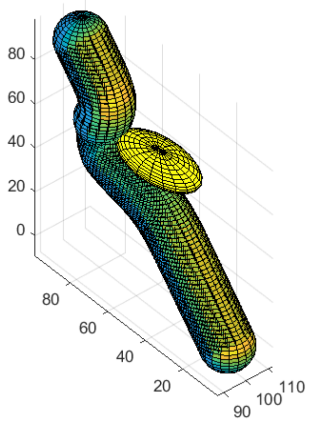
3)OBSTACLES AS ELLIPSOIDS

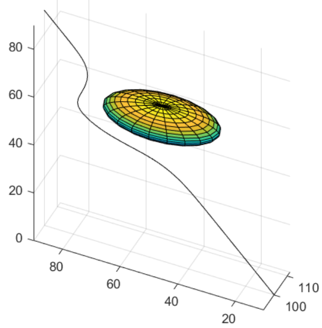




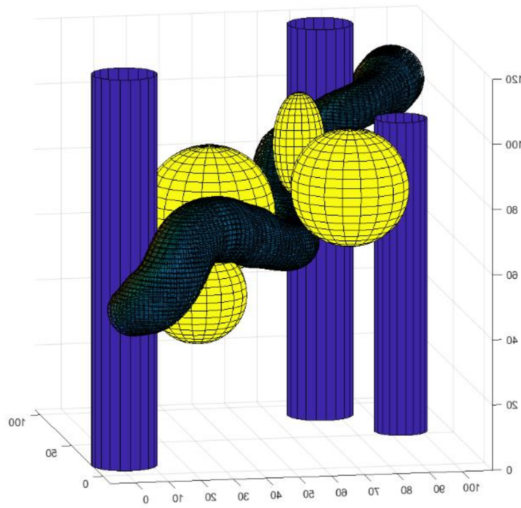


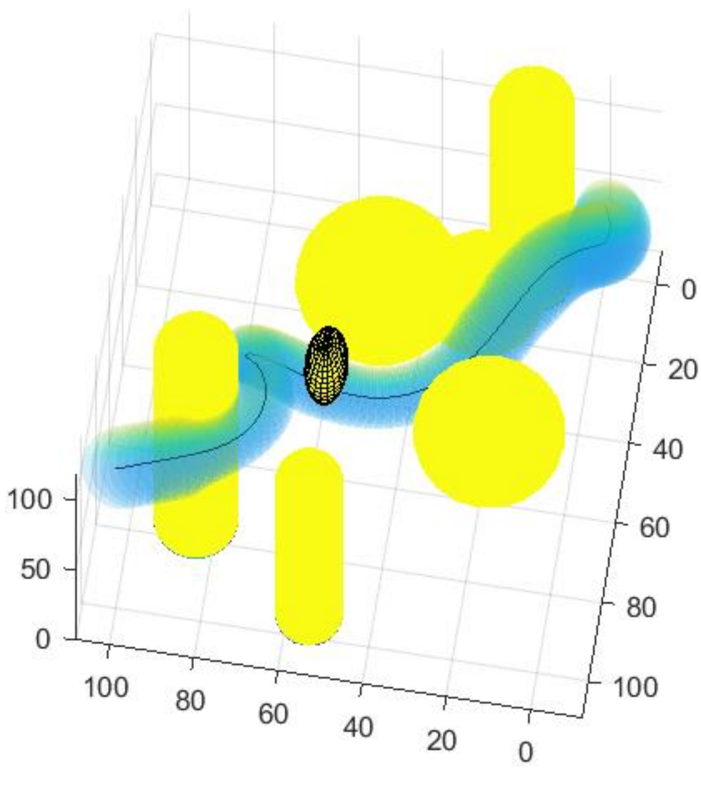


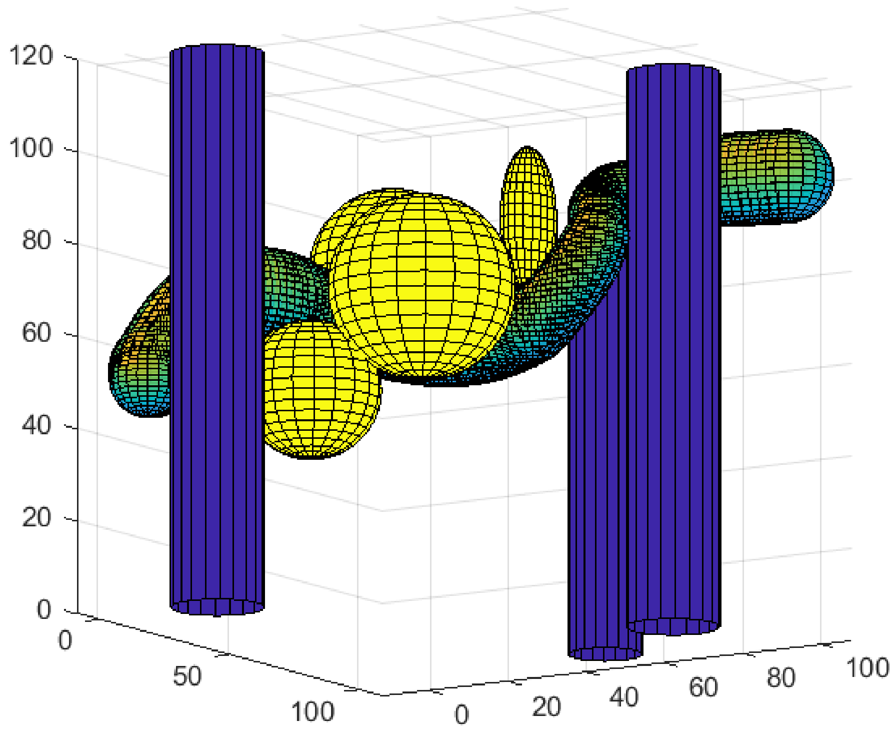


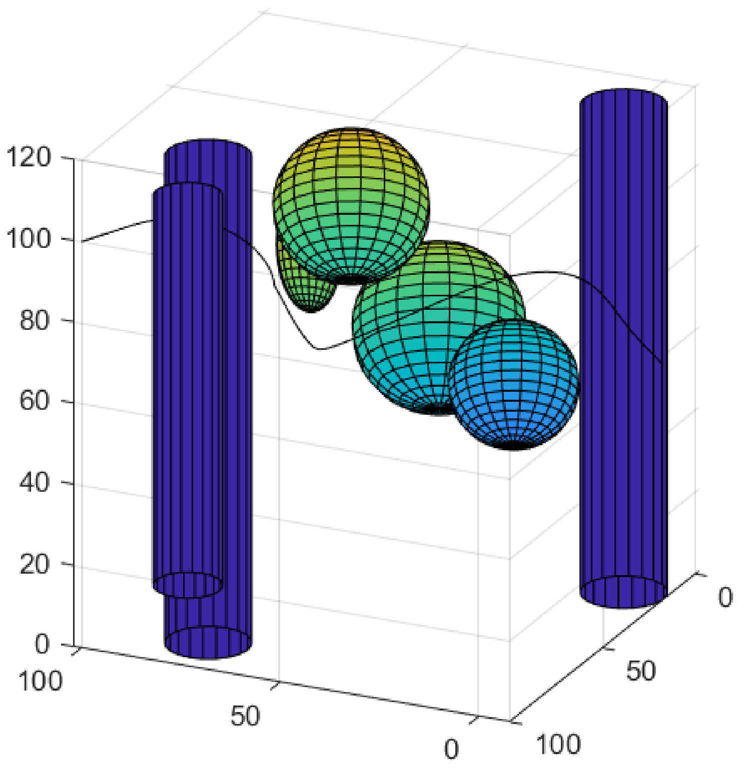


4) MIXED OBSTACLES






120
100
80
60
40
20
0
0
50
100
0
20
40
60
80
100


120
100
80
60
40
20
0
100
50
0
0
50
100

RESEARCH PAPER REFERNCES

3D Collision-Free Trajectory Generation Using Elastic Band Technique for an Autonomous Helicopter

Chi-Tai Lee[1] and Ching-Chih Tsai[2]

[1] Aeronautical Systems Research Division,
Chung-Shan Institute of Science & Technology,
Taiwan, R.O.C.
ChiTai.Lee@gmail.com
[2] Department of Electrical Engineering,
National Chung-Hsing University,
Taichung, Twiwan, R.O.C.
cctsai@nchu.edu.tw

Abstract. A real-time path generation based on the elastic band technique is presented to find a collision-free trajectory for an autonomous small-scale helicopter flying through cluttered, dynamic three-dimensional (3D) environments. The dynamic path is followed by the adaptive trajectory tracking controller augmented with the radial basis function neural networks (RBFNN). The effectiveness and merit of the proposed method are exemplified by performing three simulation scenarios: static obstacle avoidance, dynamic obstacle avoidance and terrain following.

Keywords: elastic band, obstacle avoidance, path generation, RBFNN.

https://link.springer.com/chapter/10.1007/978-3-642-23147-6_5

MATLAB CODE

```
clear all
r_min=1.5;
r_max=10;
xi = [0;0;50];
xf = [100;100;100];
N_bubbles=floor(norm(xf-xi)/r_min);
u = (xf-xi)/norm(xf-xi);                              % unit vector , p1
to p2
d = (0:norm(xf-xi)/N_bubbles:norm(xf-xi));            % displacement from
p1, along u
path = xi + u*d ;
%obstacles definition start
ox_sphere=[25,48,16];
oy_sphere=[26,34,71];
oz_sphere=[50,71,83];
o_size_sphere=[15,20,18];
ox_cylinder_new=[];
oy_cylinder_new=[];
oz_cylinder_new=[];
o_size_cylinder_new=[];
ox_cylinder=[85,55,17];
n_cylinder=1:numel(ox_cylinder);
oy_cylinder=[75,94,2];
oz_cylinder=[0,0,0];
o_size_cylinder=[10,8,10];
o_height_cylinder=[240,240,240];
for v=n_cylinder
iter=floor(o_height_cylinder(1,v)/o_size_cylinder(1,v));
ox_cylinder_new=[ox_cylinder_new,ox_cylinder(1,v)];
oy_cylinder_new=[oy_cylinder_new,oy_cylinder(1,v)];
oz_cylinder_new=[oz_cylinder_new,oz_cylinder(1,v)];
o_size_cylinder_new=[o_size_cylinder_new,o_size_cylinder(1,v)];
z_scale=o_size_cylinder(1,v);
    for y=1:iter
    ox_cylinder_new=[ox_cylinder_new,ox_cylinder(1,v)];
    oy_cylinder_new=[oy_cylinder_new,oy_cylinder(1,v)];
    oz_cylinder_new=[oz_cylinder_new,oz_cylinder(1,v)+y*z_scale];
    o_size_cylinder_new=[o_size_cylinder_new,o_size_cylinder(1,v)];
    end
end

ox_ellipsoid=[56];
oy_ellipsoid=[63];
oz_ellipsoid=[89];
o_a_ellipsoid=[5];
o_b_ellipsoid=[8];
o_c_ellipsoid=[16];
n_ellipsoid=1:numel(ox_ellipsoid);
n_sphere=1:numel(ox_sphere);
ox=[ox_sphere,ox_cylinder_new];
oy=[oy_sphere,oy_cylinder_new];
oz=[oz_sphere,oz_cylinder_new];
o_size=[o_size_sphere,o_size_cylinder_new];
%obstacles definition end
```

```
d_safe=r_min;
f_int=zeros([3,1]);
f_ext=zeros([3,1]);
alpha=1;
beta=0.9;
gamma=0.9;
n_obs=1:numel(ox);
ki=0.7;
ke=50;
b_radius_decide1=[];
b_radius_decide2=[];
% deformation of band
for k = 1:200
for i = 2:N_bubbles
    d1=norm(path(:,i+1)-path(:,i));
    d2=norm(path(:,i-1)-path(:,i));
    f_int=ki*(((d1-r_min)*1/d1*(path(:,i+1)-path(:,i)))+((d2-
r_min)*1/d2*(path(:,i-1)-path(:,i))));
    for j=n_obs
        b_radius_decide1(j)=abs(norm(path(:,i)-[ox(1,j);oy(1,j);oz(1,j)])-
o_size(1,j));
        %d_safe=b_radius_decide(j);
        if norm((path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]))>=d_safe
            d_aff=norm((path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]))-d_safe;
        end
        if norm((path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]))<d_safe
            d_aff=0;
        end
        f_ext=f_ext+ke*(exp(-b_radius_decide1(j))*1/(norm(path(:,i)-
[ox(1,j);oy(1,j);oz(1,j)]))*(path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]));

    end
    if n_ellipsoid>=1
    for n=n_ellipsoid
        e1=[1;0;0];
        e2=[0;1;0];
        e3=[0;0;1];
        c_th=sum((path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)]).*e1)/norm((path(:,
i)));
        s_th=sqrt(1-c_th^2);
        c_ph=cos(atan2(path(3,i)-oz_ellipsoid(1,n),path(2,i)-
oy_ellipsoid(1,n)));
        s_ph=sqrt(1-c_ph^2);
        a=o_a_ellipsoid(1,n);
        b=o_b_ellipsoid(1,n);
        c=o_c_ellipsoid(1,n);

r=(a*b*c)/(sqrt((b^2*c^2*c_th^2)+(c^2*a^2*s_th^2*c_ph^2)+(a^2*b^2*s_th^2*s_
ph^2)));
        b_radius_decide2(n)=abs(norm(path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)])-r);
        f_ext=f_ext+ke*(exp(-b_radius_decide2(n))*1/(norm(path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)]))*(path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)]));

    end
    end
    f_net=alpha*f_int+beta*f_ext;
    path(:,i)=path(:,i)+gamma*f_net;
    b_radius(i)=min([b_radius_decide1,b_radius_decide2]);
```

```
    if b_radius(i)>=r_max
       b_radius(i)=r_max;
    end
    if b_radius(i)<r_min
       b_radius(i)=r_min;
    end
    f_ext=0;
end
end
%reorganisation of band
for i=2:N_bubbles
   % if ((b_radius(i+1)+b_radius(i-1))>((norm(path(:,i)-path(:,i-
1)))+(norm(path(:,i+1)-path(:,i)))))
   %      for a=2:N_bubbles-1
   %         fprintf("%d",i);
   %      end

   % end
    if (b_radius(i)+b_radius(i-1)-0.01)<norm(path(:,i)-path(:,i-1))
        new_path=(path(:,i)+path(:,i-1))/2;
        for k=(N_bubbles+1):-1:i
            path(:,k+1)=path(:,k);
        end
        path(:,i)=new_path;
        d1=norm(path(:,i+1)-path(:,i));
        d2=norm(path(:,i-1)-path(:,i));
        f_int=ki*(((d1-r_min)*1/d1*(path(:,i+1)-path(:,i)))+((d2-
r_min)*1/d2*(path(:,i-1)-path(:,i))));
    for j=n_obs
        b_radius_decide1(j)=abs(norm(path(:,i)-[ox(1,j);oy(1,j);oz(1,j)])-
o_size(1,j));
        %d_safe=b_radius_decide(j);
        if norm((path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]))>=d_safe
            d_aff=norm((path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]))-d_safe;
        end
        if norm((path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]))<d_safe
            d_aff=0;
        end
        f_ext=f_ext+ke*(exp(-b_radius_decide1(j))*1/(norm(path(:,i)-
[ox(1,j);oy(1,j);oz(1,j)]))*(path(:,i)-[ox(1,j);oy(1,j);oz(1,j)]));

    end
    if n_ellipsoid>=1
    for n=n_ellipsoid
        e1=[1;0;0];
        e2=[0;1;0];
        e3=[0;0;1];
        c_th=sum((path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)]).*e1)/norm((path(:,
i)));
        s_th=sqrt(1-c_th^2);
        c_ph=cos(atan2(path(3,i)-oz_ellipsoid(1,n),path(2,i)-
oy_ellipsoid(1,n)));
        s_ph=sqrt(1-c_ph^2);
        a=o_a_ellipsoid(1,n);
        b=o_b_ellipsoid(1,n);
        c=o_c_ellipsoid(1,n);

r=(a*b*c)/(sqrt((b^2*c^2*c_th^2)+(c^2*a^2*s_th^2*c_ph^2)+(a^2*b^2*s_th^2*s_
ph^2)));
```

```
        b_radius_decide2(n)=abs(norm(path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)])-r);
        f_ext=f_ext+ke*(exp(-b_radius_decide2(n))*1/(norm(path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)]))*(path(:,i)-
[ox_ellipsoid(1,n);oy_ellipsoid(1,n);oz_ellipsoid(1,n)]));

    end
    end
    f_net=alpha*f_int+beta*f_ext;
    path(:,i)=path(:,i)+gamma*f_net;
     for k=(N_bubbles):-1:i
            b_radius(k+1)=b_radius(k);
     end
    b_radius(i)=min([b_radius_decide1,b_radius_decide2]);
    if b_radius(i)>r_max
        b_radius(i)=r_max;
    end
    if b_radius(i)<r_min
        b_radius(i)=r_min;
    end
    f_ext=0;
    N_bubbles=N_bubbles+1;
    end
end
%plotting
for l=n_cylinder
[x1,y1,z1]=cylinder;
z1(2,:)=o_height_cylinder(l)/20;
surf(x1*o_size_cylinder(l)+ox_cylinder(l),y1*o_size_cylinder(l)+oy_cylinder
(l),z1*o_size_cylinder(l)+oz_cylinder(l));
hold on;
end
for l=n_sphere
[x2,y2,z2]=sphere;
surf(x2*o_size_sphere(l)+ox_sphere(l),y2*o_size_sphere(l)+oy_sphere(l),z2*o
_size_sphere(l)+oz_sphere(l));
hold on;
end

grid on;
axis equal;
plot3(path(1,:),path(2,:),path(3,:),'color','k');
grid on;
hold on;

for k=1:N_bubbles
  [x2,y2,z2]=sphere;

h=surfl(x2*b_radius(k)+path(1,k),y2*b_radius(k)+path(2,k),z2*b_radius(k)+pa
th(3,k));
   set(h, 'FaceAlpha', 0.9);
    shading interp;
    hold on;
end

for l=n_ellipsoid
[x3,y3,z3]=ellipsoid(ox_ellipsoid(1,l),oy_ellipsoid(1,l),oz_ellipsoid(1,l),
o_a_ellipsoid(1,l),o_b_ellipsoid(1,l),o_c_ellipsoid(1,l));
surf(x3,y3,z3);
axis equal;
hold on;
```