```
function [Nei_agent, Nei_beta_agent, p_ik, q_ik, A] = findneighbors5(nodes_old,
    nodes, r, r_prime, obstacles, Rk, n, delta_t_update)

%This function is to find alpha and beta neighbors
%Created by Hung Manh La (Jan 2008)
%Oklahoma State University
%Copyright @2008 (any reproduction or modification of this code needs
    permission from the author)
%This code is not published in any paper yet. Hence if you have any
%question of this code please contact the author:lamanhhungosu@gmail.com

% Inputs:  positions of nodes (nodes),
          %active range for alpha agents(r)
          %active range for beta agents(r_prime)
          %positions of obstacles (obstacle)
          %radius of obstacles (Rk)
          %number of dimensions (n)
          %velocities of nodes (p_nodes)
          %time update (delta_t_update)

% Outputs: indices of alpha neighbors (Nei_agent)
           %indices of beta neighbors (Nei_beta_agent)
           %positions of virtual beta agent(q_ik)
           %Velocities of virtual beta agents (p_ik)


          %*******Find neighbors of alpha agent*************

num_nodes = size(nodes,1);
dif = cell(num_nodes,1); % save the difference between each alpha agent and all
    other nodes
                         % each element of cell is a matrix(size:num_nodes x n)
distance_alpha = zeros(num_nodes,num_nodes); % save the distance (norm) between
    each agent and all other nodes
                                % each column for one node
Nei_agent = cell(num_nodes,1); %Save the indices of neighbors of each agent
                               %each element of cell is a matrix (maximum size
                                   num_nodes x 1)

for i = 1:num_nodes
    dif{i} = repmat(nodes(i,:),num_nodes,1) - nodes;
    tmp = dif{i}; %recall cell i th of dif
    for j = 1:num_nodes
        d_tmp(j,:) = norm(tmp(j,:)); %compute distance between each alpha agent
            and all other nodes
    end
    distance_alpha(i,:)= d_tmp;
end

for k = 1:num_nodes
    Nei_agent{k} = find(distance_alpha(:,k)<r & distance_alpha(:,k)~=0); %find
        the neighbors of agent i
end
```

```
   %===========================================================
%*******Find neighbors of beta agent (q_ik)- Virtual beta agent********

num_obstacles = size(obstacles,1); %find number of obstacles

dif_qi_yk = cell(num_nodes,1); % save the difference between all centers of
    obstacles and each node
                               % each element of cell is a matrix (l x n), l =
                                   number of obstacles
 % Compute miu and projection matrix (ak)
miu = zeros(num_obstacles,num_nodes);% Each column for each obstacle
ak = cell(num_nodes,1); %each element of cell is matrix (num_obstacles x n)
P = cell(num_nodes,1); %each element of cell is matrix (n x n), n = 2
%Compute positions of q_ik and velocities of p_ik
q_ik = cell(num_nodes,1);%each element of cell is matrix (l x n), n= 2
p_ik = cell(num_nodes,1);%each element of cell is matrix (l x n), n= 2
dif_beta =cell(num_nodes,1); % save the difference between all beta agents and
    each node
                             % each element is a matrix(size:l x n)
distance_beta = zeros(num_obstacles,num_nodes); % save the distance (norm)
    between each shadow on obstacle
                             %and all other nodes each
                             %column for one node
Nei_beta_agent = cell(num_nodes,1);

for i = 1:num_nodes
    %Find the difference between each center of obstacle and nodes
    dif_qi_yk{i} = repmat(nodes(i,:),num_obstacles,1)- obstacles;
    tmp_dif = dif_qi_yk {i}; %recall cell i th of dif_qi_ki

    ak_tmp = zeros(num_obstacles,n); %temporary save projection matrix in each
        obstacle k
        for k = 1:num_obstacles
            miu_tmp(:,k)= Rk(k,:)/norm(tmp_dif(k,:)); %compute distance between
                qi and yk
            ak_tmp(k,:) =  tmp_dif(k,:)/norm(tmp_dif(k,:));
        end
    miu(:,i)= miu_tmp;
    ak{i}= ak_tmp;
    P{i} = eye(n,n)- (ak{i,:})'*(ak{i,:}); %Compute projection matrix (nho
        check lai)

    %++++++++++++++++++++++++++++++++++++++++
    q_ik_tmp = zeros(num_obstacles, n);%temporary save q_ik in each obstacle k
    p_ik_tmp = zeros(num_obstacles, n);%temporary save q_ik in each obstacle k
    p_i = (nodes(i,:)- nodes_old(i,:))/delta_t_update;

        for k = 1:num_obstacles
            %%Compute positions of q_ik
            q_ik_tmp(k,:) = miu(k,i)*nodes(i,:)+ (1-miu(k,i))*obstacles(k,:);
            %Compute velocities of p_ik
            p_ik_tmp(k,:) = (miu(k,i)*P{i}*p_i')';
        end

    q_ik{i} = q_ik_tmp ;
    p_ik{i} = p_ik_tmp ;
```

```
    %++++++++++++++++++++++++++++++++++++++++
     %Find beta neighbors of alpha agent after obtaining q_ik
    dif_beta{i}= q_ik{i}- repmat(nodes(i,:),num_obstacles,1);

    %Compute norm (distance) of dif_beta
    tmp_norm = dif_beta{i};
       for k = 1:num_obstacles
        distance_beta_tmp(k,:)= norm(tmp_norm(k,:));

       end
    distance_beta(:,i) = distance_beta_tmp;
    % Find neighbors
    Nei_beta_agent{i} = find(distance_beta(:,i)<r_prime & distance_beta(:,i)~=0
        ); %find the beta neighbors of agent i

end

%++++++++++++++++BUILDING THE ADJACENCY MATRIX++++++++++++++++++++

A = zeros(num_nodes,num_nodes);

for i = 1:num_nodes
    for j = 1:num_nodes
        if i ~= j
            dist_2nodes = norm(nodes(j,:) - nodes(i,:));
            if dist_2nodes < r && dist_2nodes ~= 0
                A(i,j) = 1;
            end
        end
    end
end
```