Contents


```
function [coordix, coordiy, coorditheta] = Rescheduler_gw(excluded_robot_id, robots)
```

Parameter Load

```
load('FinalResults_gw','unique_all_vertex','dmat','ccoordix','ccoordiy','idx');
load('Parameter_gw','binaryImage','gx','gy','Rangeconstant','max_value','L_range','L_range_base','RGB','RGB2','POI','unique_all_vertex_base');
```

1=simul, 0=real

```
check_debug = 1;
coordix = ccoordix;
coordiy = ccoordiy;
idxx = idx;
```

Image Load (SLAM Map)

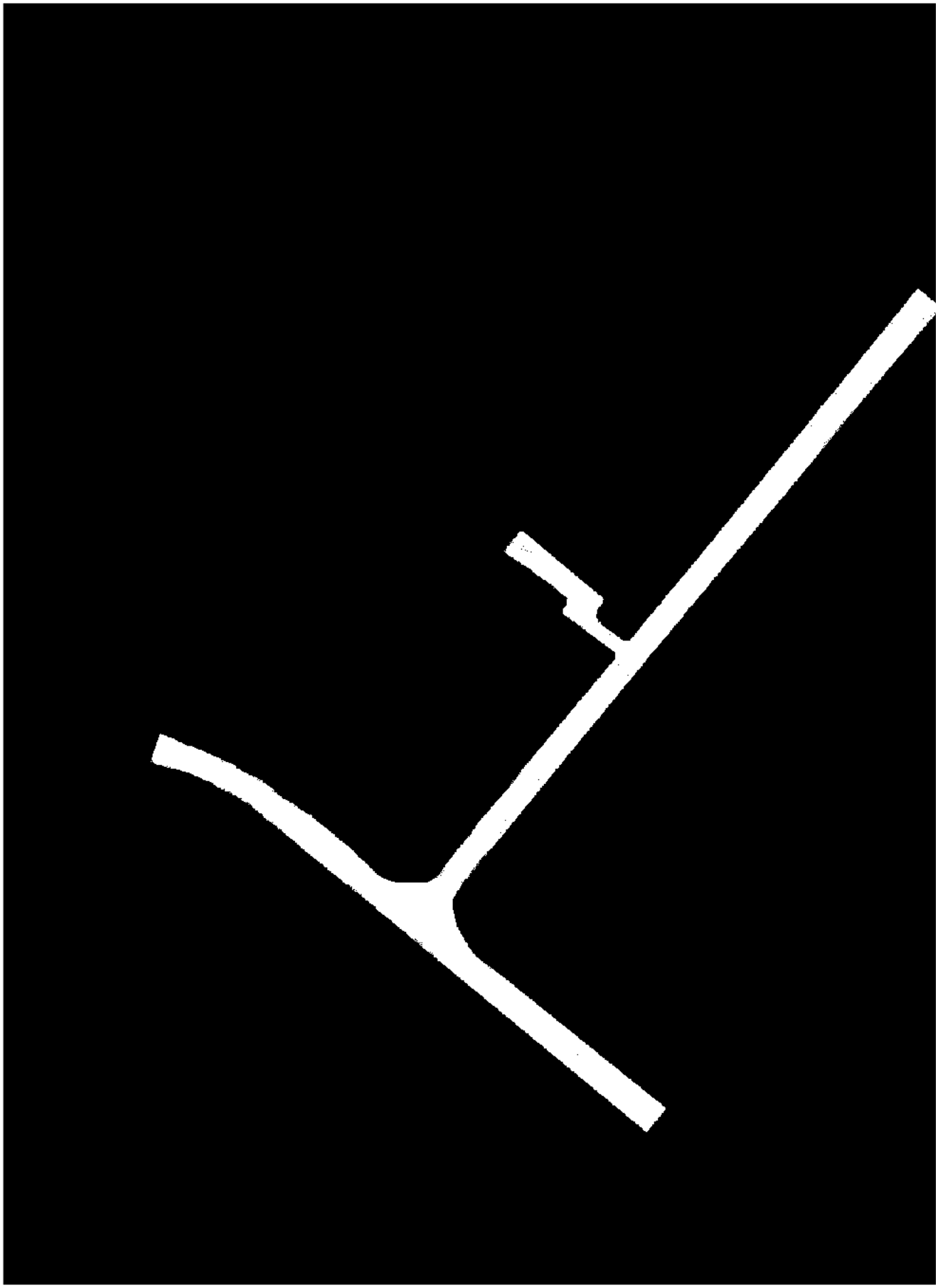
```
binaryImage = Image_load2(2.0); %% image load.최신 지도 (아직 수정필요)

fprintf('Image load completed\n');
binaryImage(binaryImage<200)=0;
binaryImage(binaryImage>=200)=255;
```

```
Image load completed
```

represent the image

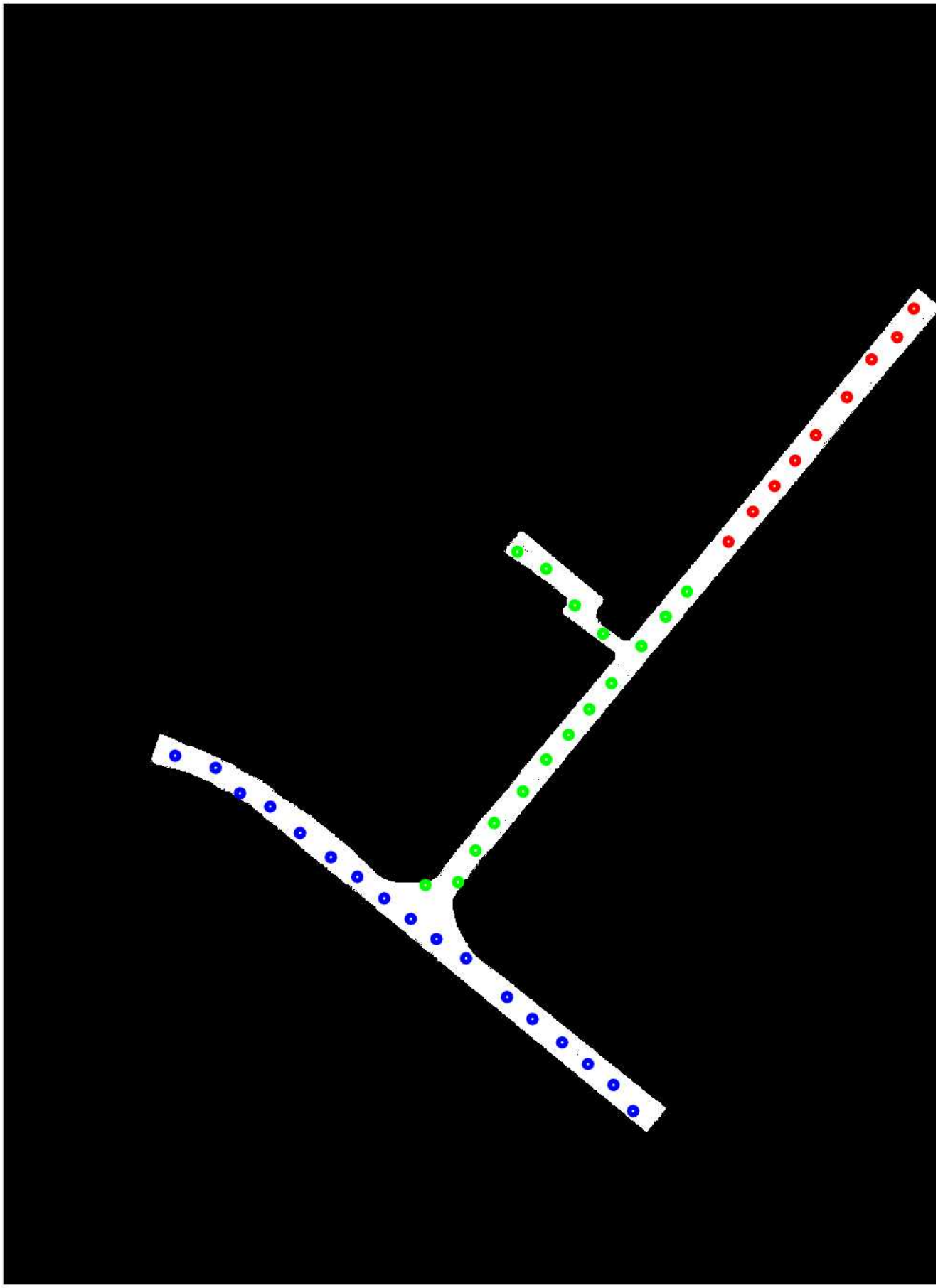
```
imshow(binaryImage );
hold on;
```



Draw Scheduler Results

```
for j=1:1000000
            count = 1;
            if(POI == 1)
                if(j < size(POI_y,2)+1)
```

```
                    plot(POI_y(j),POI_x(j),'Marker','o','color',[1 0 1],'LineWidth',4);
                end
            end
           for k = 1:size(coordiy,2)
           if(j < size(coordiy{k},2)+1 && k == 1)
              plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 0 0],'LineWidth',4);
              count = 2;
           end
            if(j < size(coordiy{k},2)+1 && k == 2)
                    plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[0 1 0],'LineWidth',4);
                    count = 2;
            end
             if(j < size(coordiy{k},2)+1 && k == 3)
               plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[0 0 1],'LineWidth',4);
               count = 2;
             end
            if(j < size(coordiy{k},2)+1 && k == 4)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[0 1 1],'LineWidth',4);
                count = 2;
             end
            if(j < size(coordiy{k},2)+1 && k ==5)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 1 0],'LineWidth',4);
                count = 2;
             end
            if(j < size(coordiy{k},2)+1 && k == 6)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 0 1],'LineWidth',4);
                count = 2;
            end
            if(j < size(coordiy{k},2)+1 && k >=7)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 1 1].*rand(1,3),'LineWidth',4);
                count = 2;
            end
           end
            if(count == 1)
                break;
            end
end
```



총 로봇 대수

for simul

```
if(check_debug)
```

```
    num_of_robots = 3;%size(robots,1); %% for simul
%     excluded_robot_id = 1; %% 2는 잘됨
 excluded_robot_id =2; %% 2,3은 잘됨
```

for Real

```
else
    num_of_robots = size(robots,1); %% for real
end
robotlist = [];
D_RobotNum = size(coordix,2);
start_idx = zeros(D_RobotNum,2);
```

이전 waypoint, 현재 waypoint index 찾기 (경로 할당시 이전 로봇 위치 고려 경로 재생성)

```
 if(check_debug)
```

for simul

```
    for i = 1:D_RobotNum
        if(i == excluded_robot_id)
            continue;
        end
        if(isempty(coordix{i}))
            continue;
        end

            start_idx(i,1) = idxx{i}(1); %% for simul
            start_idx(i,2) = idxx{i}(2);%% for simul
    end
```

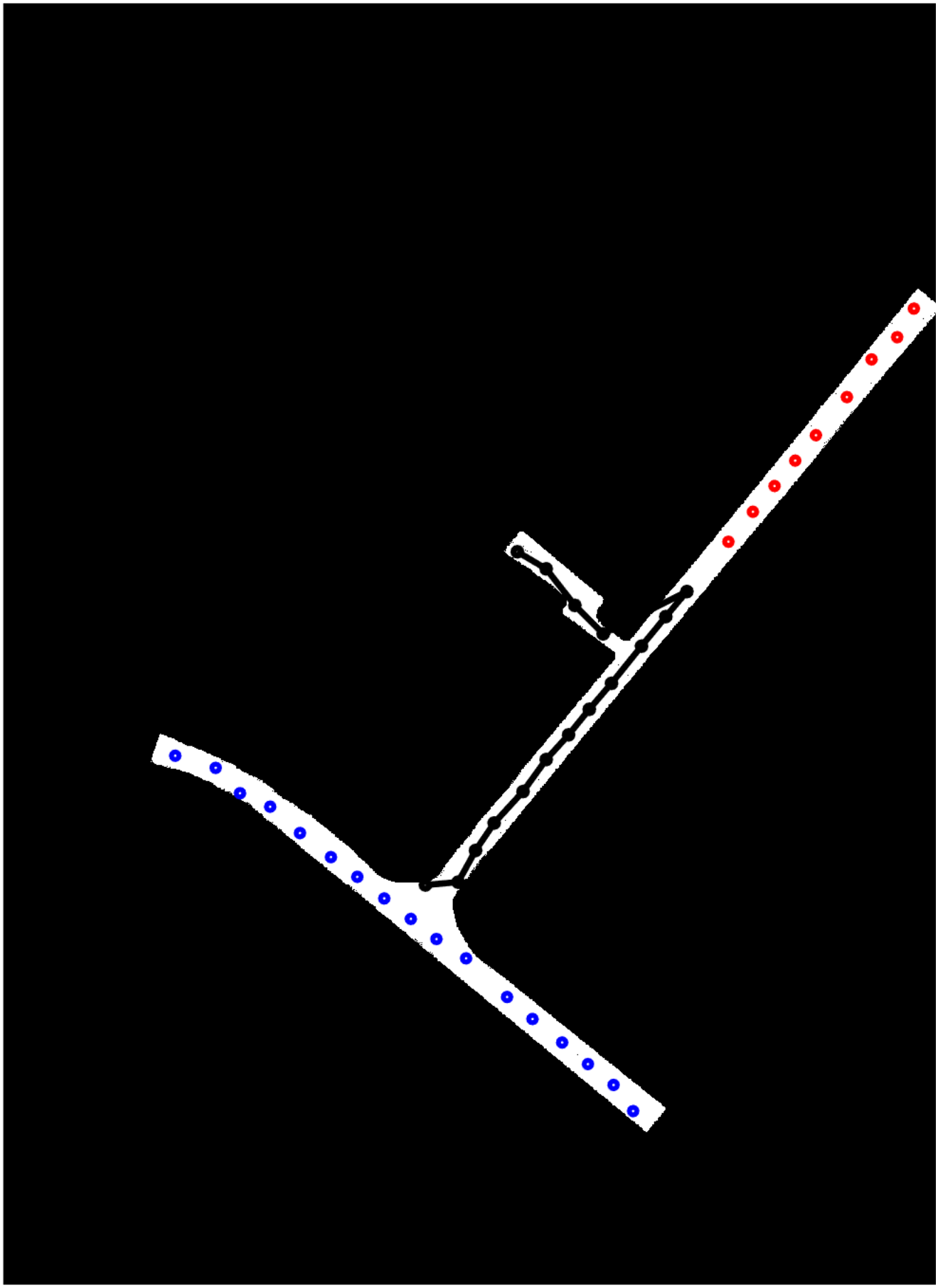
```
 else
```

for Real

```
    for i = 1:num_of_robots
          [tndist,tnidx] = sort(sqrt((wx_to_xi_gj(robots(i).curTargetx) -  unique_all_vertex(:,1)).^2+(wy_to_yi_gj(robots(i).curTargety) -  unique_all_vertex(:,2)).^2));
          [tndist1,tnidx1] =  sort(sqrt((wx_to_xi_gj(robots(i).prevTargetx) -  unique_all_vertex(:,1)).^2+(wy_to_yi_gj(robots(i).prevTargety) -  unique_all_vertex(:,2)).^2));
          if(~isempty(find(idxx{robots(i).robotId}==tnidx)) && ~isempty(find(idxx{robots(i).robotId}==tnidx1)))
              start_idx(robots(i).robotId,1) = tnidx1(1);
              start_idx(robots(i).robotId,2) = tnidx(1);
          else
               start_idx(robots(i).robotId,1) = idxx{robots(i).robotId}(1);
               start_idx(robots(i).robotId,2) = idxx{robots(i).robotId}(2);
          end
    end
```

```
 end

% figure(1)
hold on;
target_robotnum = excluded_robot_id;  %% 임의 지정
plot(coordiy{target_robotnum}(:),coordix{target_robotnum}(:),'Marker','o','color',[0 0 0],'LineWidth',5);
% pause;
s = [];
t = [];
```



거리 기반 Relational Graph 및 Tree 생성

```
for i=1:size(coordiy,2)
    for j=i:size(coordiy,2)
        if(i~=j && ~isempty(coordiy{j}) && ~isempty(coordiy{i}))
           aa = repmat(coordiy{i}(:),size(coordiy{j},2),1)- kron(coordiy{j}(:),ones(size(coordiy{i},2),1));
```

```
            bb = repmat(coordix{i}(:),size(coordix{j},2),1)- kron(coordix{j}(:),ones(size(coordix{i},2),1));
            cc = sqrt(aa.^2+bb.^2);
            if(min(cc) <= L_range*2) %% 거리 기반
                s = [s i];
                t = [t j];
            end
        end
    end
end
G=graph(s,t);
TR = shortestpathtree(G,target_robotnum,'all');
```

area_allocation => Tree의 레벨 표시 행렬

```
area_allocation = zeros(size(coordiy,2),size(coordiy,2));
TR1 = [];
for i=1:size(coordix,2)
    if(target_robotnum~=i)
        TR1 = shortestpath(G,target_robotnum,i);
        count = 2;
        while(count <= size(TR1,2))
            area_allocation(TR1(count),count-1) = area_allocation(TR1(count),count-1)+1;
            count = count+1;
        end
    end
end

fprintf('Robot Relational Graph GenerationWn');


    coorditesty = coordiy{target_robotnum}(:);
    coorditestx = coordix{target_robotnum}(:);
    idx_test = idx{target_robotnum}(:);
    assing_region_num = size(coorditesty,1)/(D_RobotNum-1);
    whole_size = size(coorditesty,1);
```

```
Robot Relational Graph Generation
```

Propagation 시작

```
for t_count = 1:D_RobotNum
```

```
    area_division_num = sum(area_allocation(:,t_count));
    area_idx = (1:area_division_num);
    if(area_division_num==0)
        break;
    end

    TR1 = shortestpath(G,target_robotnum,t_count);
```

```
Final Result Check
```

for debug, Draw Figures

```
hold on;
   for j=1:1000000
            count = 1;
           for k = 1:size(coordiy,2)
           if(j < size(coordiy{k},2)+1 && k == 1)
              plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 0 0],'LineWidth',4);
              count = 2;
           end

            if(j < size(coordiy{k},2)+1 && k == 2)
                    plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[0 1 0],'LineWidth',4);
                    count = 2;
            end
             if(j < size(coordiy{k},2)+1 && k == 3)
               plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[0 0 1],'LineWidth',4);
               count = 2;
             end
            if(j < size(coordiy{k},2)+1 && k == 4)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[0 1 1],'LineWidth',4);
                count = 2;
             end
            if(j < size(coordiy{k},2)+1 && k ==5)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 1 0],'LineWidth',4);
                count = 2;
             end
            if(j < size(coordiy{k},2)+1 && k == 6)
                plot(coordiy{k}(j),coordix{k}(j),'Marker','o','color',[1 0 1],'LineWidth',4);
                count = 2;
            end
           end
            if(count == 1)
                break;
            end
   end
```

for debug, Draw Figures

```
      fprintf('debug, Draw Figures');
%       pause;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
debug, Draw Figures
```

t_count = 1;

```
    if(t_count == 1)%
```

```
        count_region = zeros(D_RobotNum,1);%
        min_area_size = 100000;
        min_area_size_index = 0;
       for i = 1:D_RobotNum
```

region assignment

```
            count_region(i) = area_allocation(i,t_count)*floor(assing_region_num);
            if(area_allocation(i,t_count)~=0)
                add_size =size(coordiy{i},2)+count_region(i);
                if(add_size < min_area_size)
                    min_area_size = add_size;
                    min_area_size_index = i;
                end
            end
```

```
       end
```

Considerations: # of regions for the first time

```
       if(~(size(coordiy{target_robotnum},2)-sum(count_region)==0))
           count_region(min_area_size_index) = count_region(min_area_size_index)+(size(coordiy{target_robotnum},2)-sum(count_region));
       end
         assignment_region = count_region;
      assignment_region = assignment_region.*0;
      original_count_region = count_region;
       while(1)
           count_region_check = 0;
           for i=1:D_RobotNum
               if(count_region(i)<=0)
                   count_region_check = count_region_check+1;
               else
%                     TR1 = shortestpath(G,target_robotnum,i);
%                  target_robotnum1 = TR1(end-1);
                  [coordiy,coordix,idxx,assign_ok]=region_assign(coordiy,coordix,idxx,i,target_robotnum,1,L_range*2);
                  if(assign_ok)
                     count_region(i) = count_region(i)-1;
                     assignment_region(i) = assignment_region(i) + 1;
                  else
```

할당될 노드가 없으면 할당할 수 있는 로봇에 자신이 가져가야할 노드까지 할당

추가로 가져가야하지만 가지가지 못할 경우, 본인이 가져가야할 몫은 +로 두고 더가져간 로봇의 몫은 -로 표기

```
                     a = find(count_region<0);
                     b = find(original_count_region~=0);
                     b = b(b~=i);
                     c = find(~ismember(b,a));
                     d = b(c(1));
                     count_region(d) = count_region(d)+count_region(i);
                     count_region(i) = -1*count_region(i);
```

```
                  end
                     Draw(i,coordiy,coordix);
               end
           end
           if(count_region_check==D_RobotNum)

                diff = assignment_region - original_count_region;
               while(sum(abs(diff)) > 0)
                   for i = 1:D_RobotNum
                       if((original_count_region(i)-assignment_region(i))>0)
                            diff = assignment_region - original_count_region;
                            [an,or] = sort(diff);
                            for j=0:D_RobotNum-1

                                if(or(end-j)~=target_robotnum)
                                    [coordiy,coordix,idxx,assign_ok]=region_assign(coordiy,coordix,idxx,i,or(end-j),1,L_range*2);
                                      if(assign_ok)
                                          assignment_region(i) = assignment_region(i)+1;
                                          assignment_region(or(end-j)) = assignment_region(or(end-j))-1;
                                          Draw(i,coordiy,coordix);
                                          break;
```

```
                                    end
                                end
                            end
                        end
                    end
                    diff = assignment_region - original_count_region;
                end
                break;
            end
        end
    fprintf('first ifWn');
```

```
first if
```

```
    else
```

Relational Graph로부터 얻은 Tree의 자식 노드들 수행 (root 노드 제외)

```
        drawfirst = 0;
        count_region = zeros(D_RobotNum,1);
        for i = 1:D_RobotNum
             count_region(i) = area_allocation(i,t_count)*floor(assing_region_num);
        end
        while(1)
            count_region_check = 0;
            for i=1:D_RobotNum
                if(count_region(i)==0)
                    count_region_check = count_region_check+1;
                else
```

```
                    TR1 = shortestpath(G,target_robotnum,i);
                target_robotnum1 = TR1(end-1);
```

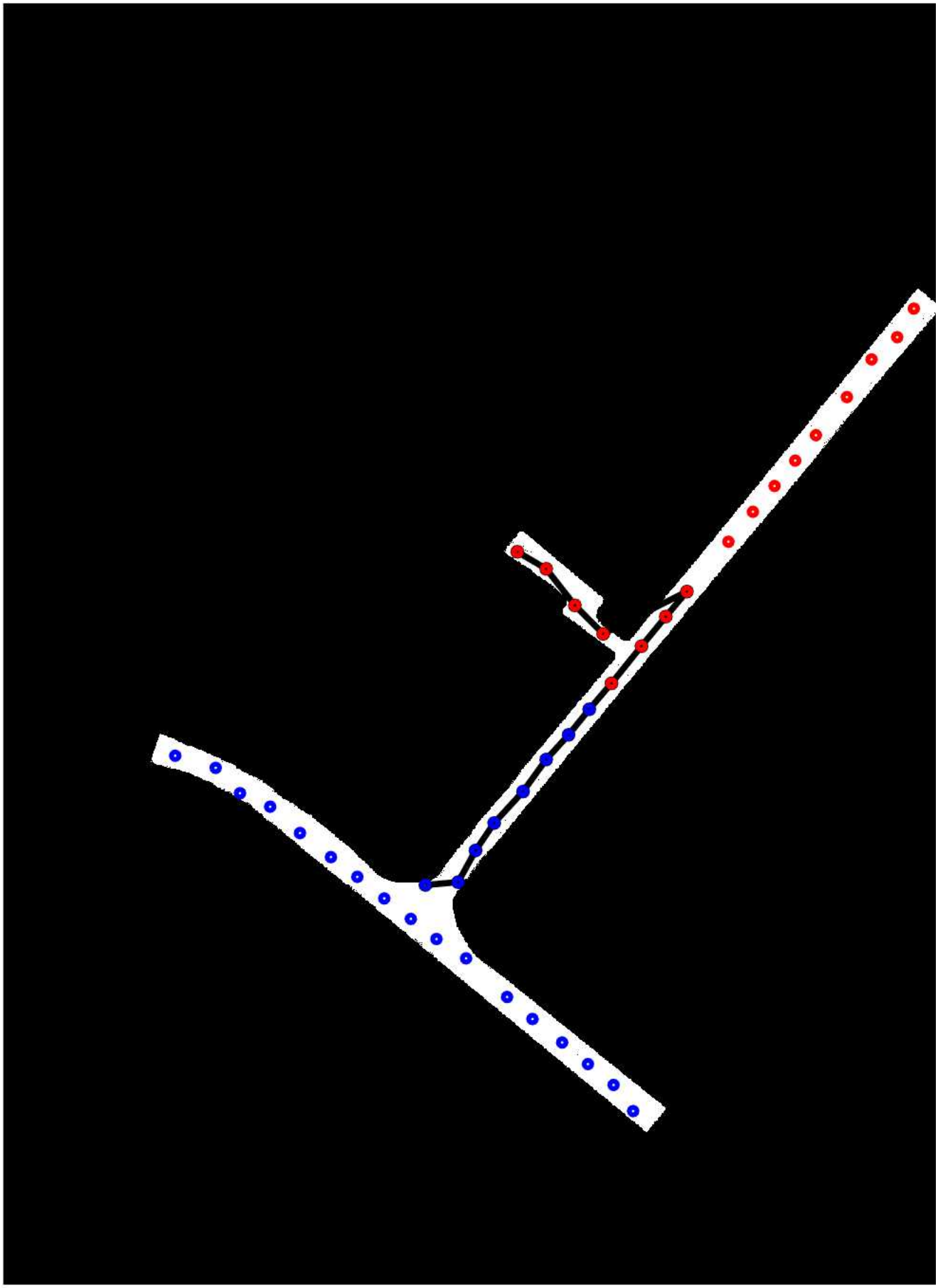
region_assign_probabilistic: kmeans 수행시 그룹간 거리합을 통해 확률적인 region reassign 수행.

```
                [coordiy,coordix,idxx]=region_assign_probabilistic(coordiy,coordix,idxx,i,target_robotnum1,1,L_range*2);
                    count_region(i) = count_region(i)-1;

                    Draw(i,coordiy,coordix);
%                     pause;
```

```
                end
            end
            if(count_region_check==D_RobotNum)
                break;
            end
        end
```

```
    end
```



```
end

coordiy{target_robotnum} = [];
coordix{target_robotnum} = [];
```

```
idxx{target_robotnum} = [];
fprintf('Final Result CheckWn');
```

Seamless Patrol Path 생성

```
        for i = 1:D_RobotNum
            if(isempty(coordiy{i}))
                continue;
            end

            [shortestPath,shortestPathLength] = re_onewaypath(binaryImage,coordix{i}(:),coordiy{i}(:),dmat,idxx{i}(:),L_range,Rangeconstant,start_idx(i,:));
            coordiy{i} = coordiy{i}(shortestPath);
            coordix{i} = coordix{i}(shortestPath);
        end
```

For real, 좌표계 변환

```
   posx= coordix;
   posy= coordiy ;
   posTheta=[];
offsetx = 3140;
offsety = 3260 ;
for i=1:size(coordix,2)
      for j = 1:size(coordix{i},2)
         posx{i}(1,j) = 0.05*(  coordiy{i}(1,j)-(offsety));
         posy{i}(1,j) = 0.05*( (offsetx) - coordix{i}(1,j));
         if(j ==size(coordix{i},2) )
             posTheta{i}(1,j) = atan2(posy{i}(1,1)-posy{i}(1,j),posx{i}(1,1)-posx{i}(1,j));
         else
             posTheta{i}(1,j) = atan2(0.05*((offsetx) - coordix{i}(1,j+1))-posy{i}(1,j),0.05*(coordiy{i}(1,j+1)-(offsety))-posx{i}(1,j));
         end
      end
end
    coordix = posx;
    coordiy = posy;
    coorditheta = posTheta;
fprintf('kw EndWn');
```

```
kw End

ans =

  1×3 cell 배열

    {1×21 double}    {0×0 double}    {1×25 double}
```

Published with MATLAB® R2020a