```
function [T, nodes] = buildRRT(X_start, X_goal, M)

    % Create NodeList
    nodes = [X_start(1) X_start(2) 0];
    pathFound = false;
    maxIterations = 10000;
    iterations = 0;
    mapRanges = getMapRanges(M);

    % Expand the tree until a path is found OR too many iterations have
    % passed.
    while iterations < maxIterations && pathFound == false

        % Choose node for expansion
        shortestd2 = 9999999;
        bTol = 2;
        randX = mapRanges(1) - bTol + rand()*(mapRanges(2)-
                                              mapRanges(1)+2*bTol);
        randY = mapRanges(3) - bTol + rand()*(mapRanges(4)-
                                              mapRanges(3)+2*bTol);
        for i=1:size(nodes,1)
            d2 = (randX - nodes(i,1))^2 + (randY - nodes(i,2))^2;
            if d2 < shortestd2
                shortestd2 = d2;
                closestNodeIndex = i;
            end
        end
        nodeToExpand = nodes(closestNodeIndex,1:3);

        % Expand Node
        randDist = 0.05 + 1.5*rand();
        randAng = 2*pi*rand();
        newNode = [nodeToExpand(1) + randDist*cos(randAng)
             nodeToExpand(2) + randDist*sin(randAng) closestNodeIndex];

        % Check for collision
        if collisionFound(nodeToExpand, newNode, mapRanges, M) == false

            % Add to nodelist
            nodes = [nodes; newNode];

            % Check for goal region
            if collisionFound(newNode, X_goal, mapRanges, M) == false
                nodes = [nodes; [X_goal(1) X_goal(2) size(nodes,1)]];
                pathFound = true;
            end
        end

        % Increment number of iterations
        iterations = iterations + 1;
    end

    % Create the trajectory to follow
    T = BuildOptimalPath(nodes, mapRanges, M);

end
```