VI APPENDIX

Develop Local Planner Program:

```
teb.ReferencePath = smoothedReferencePath;
curpose = smoothedReferencePath(1,:);
curvel = [0 0];
simtime = 0;
tsReplan = 3;
tsIntegrator = 0.001; % Reducing timestep can lead to more accurate path tracking
tsVisualize = 0.1;
itr = 0;
goalReached = false;
tVis = inf;
tPlan = inf;
adjustedPath = 0;
exampleHelperPlotLines(teb.ReferencePath,{"MarkerSize",10});
hTEBPath1_1 = quiver(nan,nan,nan,nan,.2,DisplayName="Current Path");
figure;
move(localMap,curpose(1:2),MoveType="Absolute");
syncWith(localMap,obstacles);
h = show(localMap);
ax2 = h.Parent;
hold on;
exampleHelperPose2Quiver(originalReferencePath,{"AutoScale","off"});
```

```
exampleHelperPose2Quiver(smoothedReferencePath,{"AutoScale","off"});
hRef = exampleHelperPlotLines(teb.ReferencePath,{"MarkerSize",10});
hTEBPath1_2 = quiver(nan,nan,nan,nan,.2,DisplayName="Current Path");
[~,hVeh] = exampleHelperCreateVehicleGraphic(gca,"Start",collisionChecker);
hTEBPath2_2 = hgtransform;
arrayfun(@(x)set(x,"Parent",hTEBPath2_2),hVeh);
hRefCur = exampleHelperPlotLines(teb.ReferencePath,".-");
while norm(curpose(1:2) - smoothedReferencePath(end,1:2),2) > 10


    if tVis >= tsVisualize
        move(localMap,curpose(1:2),"MoveType","Absolute","SyncWith",obstacles);
        show(localMap,Parent=ax2,FastUpdate=1);
        hTEBPath2_2.Matrix(1:3,:) = [eul2rotm([0 0 curpose(3)],'XYZ')
[curpose(1:2)';0]];
        drawnow limitrate;
        tVis = 0;
    end
    if tPlan >= tsReplan

move(localMap,curpose(1:2),"MoveType","Absolute","SyncWith",obstacles);
         [velcmds,tstamps,curpath,info] = teb(curpose, curvel);
        if info.HasReachedGoal
            break;
        end
        set(hTEBPath1_1,XData=curpath(:,1),YData=curpath(:,2), ...
```

```
            UData=cos(curpath(:,3)),VData=sin(curpath(:,3)));
        set(hTEBPath1_2,XData=curpath(:,1),YData=curpath(:,2), ...
            UData=cos(curpath(:,3)),VData=sin(curpath(:,3)));
        set(hRefCur,XData=teb.ReferencePath(:,1),YData=teb.ReferencePath(:,2));
        hTEBPath2_2.Matrix(1:3,:) = [eul2rotm([0 0 curpose(3)],'XYZ')
[curpose(1:2)';0]]
            teb.RobotInformation.Dimension(1),teb.RobotInformation.Dimension(2));
        if needLocalReplan
            continue;
        else
            if needFreeSpaceReplan
                error('Need replan');
            end
        end
        timestamps = tstamps + simtime;
        tVis = 0;
        tPlan = 0;
    end
        adjustedPath = 0;
    simtime = simtime + tsIntegrator;
    tVis = tVis + tsIntegrator;
    tPlan = tPlan+tsIntegrator;
    velcmd = velocityCommand(velcmds, timestamps, simtime);
    statedot = [velcmd(1)*cos(curpose(3)) ...
                velcmd(1)*sin(curpose(3)) ...
```

```
            velcmd(2)];
    curpose = curpose + statedot * tsIntegrator;
    curvel = velcmd;
end
```

TERRAIN-AWARE FOR OFFROAD NAVIGATION PROGRAM

```
Create map with resolution of 1 cell per meter
res = 1;
binMap = binaryOccupancyMap(~imSlope,res);
localPathList = pathList;
for i = 1:numel(pathList)
    localPathList(i).Path = grid2local(binMap,pathList(i).Path);
end
maxElementPerEdge = 50;
[nodes,edges,edge2pathIdx,cachedPaths] =
exampleHelperPath2GraphData(localPathList,maxElementPerEdge);
edgeCosts = exampleHelperDefaultEdgeCost(cachedPaths,edge2pathIdx);
stateTable = table(nodes,VariableNames={'StateVector'});
linkTable =
table(edges,edgeCosts,edge2pathIdx(:),VariableNames={'EndStates','Weight','Edg
e2PathIdx'});
roadNetwork = navGraph(stateTable,linkTable);
routePlanner = plannerAStar(roadNetwork);
start = [286.5 423.5 -pi/2];
goal  = [795.5 430.5  pi];
```

```
[dStart,nearStartIdx] = min(vecnorm(nodes(:,1:2)-start(1:2),2,2));

[dGoal,nearGoalIdx] = min(vecnorm(nodes(:,1:2)-goal(1:2),2,2));

[waypoints, solnInfo] = plan(routePlanner,nearStartIdx,nearGoalIdx);

edgePairs = [solnInfo.PathStateIDs(1:end-1)' solnInfo.PathStateIDs(2:end)'];

linkID = findlink(routePlanner.Graph,edgePairs);

networkPath =
vertcat(cachedPaths(routePlanner.Graph.Links.Edge2PathIdx(linkID)).Path);

subsampleInterval = 5;

smoothedNetworkPath =
exampleHelperSmoothPath(networkPath,subsampleInterval);

hIm = show(binMap);

hold on
```

Run Simulation

Develop Local Planner

```
load("OpenPitMinePart1Data.mat","dem","pathList");

load("OpenPitMinePart2Data.mat","originalReferencePath","smoothedReferenceP
ath","fixedTerrainAwareParams","tuneableTerrainAwareParams");
[costMap,maxSlope] =
exampleHelperDem2mapLayers(dem,tuneableTerrainAwareParams.MaxAngle,fix
edTerrainAwareParams.Resolution);

obstacles = getLayer(costMap,"terrainObstacles");

[tunableTEBParams,fixedTEBParams] = exampleHelperTEBParams

vehDims =
exampleHelperVehicleGeometry(fixedTEBParams.Length,fixedTEBParams.Width
,"collisionChecker");

collisionChecker = inflationCollisionChecker(vehDims,3);

exampleHelperInflateRoadNetwork(obstacles,pathList,collisionChecker.InflationR
adius*1.5);
```

```
maxDistance =
(tunableTEBParams.MaxVelocity(1)*tunableTEBParams.LookaheadTime/obstacl
es.Resolution);

localMap =
binaryOccupancyMap(2*maxDistance,2*maxDistance,obstacles.Resolution);

localMap.GridOriginInLocal = -localMap.GridSize/(2*localMap.Resolution);

teb = controllerTEB(smoothedReferencePath, localMap);

teb.NumIteration            = fixedTEBParams.NumIteration;

teb.ReferenceDeltaTime      = fixedTEBParams.ReferenceDeltaTime;

teb.RobotInformation        = fixedTEBParams.RobotInformation;

teb.ObstacleSafetyMargin    = collisionChecker.InflationRadius*2;

teb.LookAheadTime           = tunableTEBParams.LookaheadTime; % In seconds

teb.CostWeights.Time        = tunableTEBParams.CostWeights.Time;

teb.CostWeights.Smoothness  = tunableTEBParams.CostWeights.Smoothness;

teb.CostWeights.Obstacle    = tunableTEBParams.CostWeights.Obstacle;

teb.MinTurningRadius        = tunableTEBParams.MinTurningRadius;

teb.MaxVelocity             = tunableTEBParams.MaxVelocity;

teb.MaxAcceleration         = tunableTEBParams.MaxAcceleration;

teb.MaxReverseVelocity      = tunableTEBParams.MaxReverseVelocity;

teb.ReferencePath = smoothedReferencePath;

curpose = smoothedReferencePath(1,:);

curvel = [0 0];

simtime = 0;

tsReplan = 3;

tsIntegrator = 0.001; % Reducing timestep can lead to more accurate path tracking

tsVisualize = 0.1;
```

```
itr = 0;
goalReached = false;
tVis = inf;
tPlan = inf;
adjustedPath = 0;
hold on
exampleHelperPlotLines(teb.ReferencePath,{"MarkerSize",10});
hTEBPath1_1 = quiver(nan,nan,nan,nan,.2,DisplayName="Current Path");
figure;
move(localMap,curpose(1:2),MoveType="Absolute");
syncWith(localMap,obstacles);
h = show(localMap);
ax2 = h.Parent;
hold on;
exampleHelperPose2Quiver(originalReferencePath,{"AutoScale","off"});
exampleHelperPose2Quiver(smoothedReferencePath,{"AutoScale","off"});
hRef = exampleHelperPlotLines(teb.ReferencePath,{"MarkerSize",10});
hTEBPath1_2 = quiver(nan,nan,nan,nan,.2,DisplayName="Current Path");
[~,hVeh] = exampleHelperCreateVehicleGraphic(gca,"Start",collisionChecker);
hTEBPath2_2 = hgtransform;
arrayfun(@(x)set(x,"Parent",hTEBPath2_2),hVeh);
hRefCur = exampleHelperPlotLines(teb.ReferencePath,".-");
while norm(curpose(1:2) - smoothedReferencePath(end,1:2),2) > 10
    if tVis >= tsVisualize
        move(localMap,curpose(1:2),"MoveType","Absolute","SyncWith",obstacles);
```

```
        show(localMap,Parent=ax2,FastUpdate=1);
        hTEBPath2_2.Matrix(1:3,:) = [eul2rotm([0 0 curpose(3)],'XYZ')
[curpose(1:2)';0]];
        drawnow limitrate;
        tVis = 0;
    end
    if tPlan >= tsReplan
        move(localMap,curpose(1:2),"MoveType","Absolute","SyncWith",obstacles);
              [velcmds,tstamps,curpath,info] = teb(curpose, curvel);
        if info.HasReachedGoal
            break;
        end
              set(hTEBPath1_1,XData=curpath(:,1),YData=curpath(:,2), ...
             UData=cos(curpath(:,3)),VData=sin(curpath(:,3)));
        set(hTEBPath1_2,XData=curpath(:,1),YData=curpath(:,2), ...
             UData=cos(curpath(:,3)),VData=sin(curpath(:,3)));
        set(hRefCur,XData=teb.ReferencePath(:,1),YData=teb.ReferencePath(:,2));
        hTEBPath2_2.Matrix(1:3,:) = [eul2rotm([0 0 curpose(3)],'XYZ')
[curpose(1:2)';0]];

        if needLocalReplan
            continue;
        else
            if needFreeSpaceReplan
                error('Need replan');
```

```
        end
    end
    timestamps = tstamps + simtime;
    tVis = 0;
    tPlan = 0;
end
adjustedPath = 0;
simtime = simtime + tsIntegrator;
tVis = tVis + tsIntegrator;
tPlan = tPlan+tsIntegrator;
velcmd = velocityCommand(velcmds, timestamps, simtime);
    statedot = [velcmd(1)*cos(curpose(3)) ...
            velcmd(1)*sin(curpose(3)) ...
            velcmd(2)];
curpose = curpose + statedot * tsIntegrator;
curvel = velcmd;
end
```

SIMULATING AUTONOMOUS VEHICLE ON A SAMPLE TERRAIN

```
load("OpenPitMinePart1Data.mat","dem","pathList");
 [costMap,maxSlope] =
exampleHelperDem2mapLayers(dem,tuneableTerrainAwareParams.MaxAngle,fix
edTerrainAwareParams.Resolution);obstacles =
getLayer(costMap,"terrainObstacles");
```

```
[tunableTEBParams,fixedTEBParams] = exampleHelperTEBParams

vehDims =
exampleHelperVehicleGeometry(fixedTEBParams.Length,fixedTEBParams.Width
,"collisionChecker");

collisionChecker = inflationCollisionChecker(vehDims,3);

maxDistance =
(tunableTEBParams.MaxVelocity(1)*tunableTEBParams.LookaheadTime/obstacl
es.Resolution);

localMap =
binaryOccupancyMap(2*maxDistance,2*maxDistance,obstacles.Resolution);

localMap.GridOriginInLocal = -localMap.GridSize/(2*localMap.Resolution);

teb = controllerTEB(smoothedReferencePath, localMap);

teb.NumIteration           = fixedTEBParams.NumIteration;

teb.ReferenceDeltaTime     = fixedTEBParams.ReferenceDeltaTime;

teb.RobotInformation       = fixedTEBParams.RobotInformation;

teb.ObstacleSafetyMargin   = collisionChecker.InflationRadius*2;

teb.LookAheadTime          = tunableTEBParams.LookaheadTime; % In seconds

teb.CostWeights.Time       = tunableTEBParams.CostWeights.Time;

teb.CostWeights.Smoothness  = tunableTEBParams.CostWeights.Smoothness;

teb.CostWeights.Obstacle   = tunableTEBParams.CostWeights.Obstacle;

teb.MinTurningRadius       = tunableTEBParams.MinTurningRadius;

teb.MaxVelocity            = tunableTEBParams.MaxVelocity;

teb.MaxAcceleration        = tunableTEBParams.MaxAcceleration;

teb.MaxReverseVelocity     = tunableTEBParams.MaxReverseVelocity;

No output or LastFeasibleIdx == 1: May occur alongside any ExitFlag

teb.ReferencePath = smoothedReferencePath;

curpose = smoothedReferencePath(1,:);
```

```
curvel = [0 0];
simtime = 0;
tsReplan = 3;
tsIntegrator = 0.001; % Reducing timestep can lead to more accurate path tracking
tsVisualize = 0.1;
itr = 0;
goalReached = false;
tVis = inf;
tPlan = inf;
adjustedPath = 0;
hold on
move(localMap,curpose(1:2),MoveType="Absolute");
syncWith(localMap,obstacles);
h = show(localMap);
ax2 = h.Parent;
hold on;
exampleHelperPose2Quiver(originalReferencePath,{"AutoScale","off"});
exampleHelperPose2Quiver(smoothedReferencePath,{"AutoScale","off"});
hRef = exampleHelperPlotLines(teb.ReferencePath,{"MarkerSize",10});
hTEBPath1_2 = quiver(nan,nan,nan,nan,.2,DisplayName="Current Path");
[~,hVeh] = exampleHelperCreateVehicleGraphic(gca,"Start",collisionChecker);
hTEBPath2_2 = hgtransform;
arrayfun(@(x)set(x,"Parent",hTEBPath2_2),hVeh);
hRefCur = exampleHelperPlotLines(teb.ReferencePath,".-");
while norm(curpose(1:2) - smoothedReferencePath(end,1:2),2) > 10
```

```
    if tVis >= tsVisualize

        move(localMap,curpose(1:2),"MoveType","Absolute","SyncWith",obstacles);

        show(localMap,Parent=ax2,FastUpdate=1);

        hTEBPath2_2.Matrix(1:3,:) = [eul2rotm([0 0 curpose(3)],'XYZ')
[curpose(1:2)';0]];

        drawnow limitrate;

        tVis = 0;

    end

    if tPlan >= tsReplan

        move(localMap,curpose(1:2),"MoveType","Absolute","SyncWith",obstacles);

              [velcmds,tstamps,curpath,info] = teb(curpose, curvel);

        if info.HasReachedGoal

           break;

        end

              set(hTEBPath1_1,XData=curpath(:,1),YData=curpath(:,2), ...

             UData=cos(curpath(:,3)),VData=sin(curpath(:,3)));

        set(hTEBPath1_2,XData=curpath(:,1),YData=curpath(:,2), ...
```

```
            UData=cos(curpath(:,3)),VData=sin(curpath(:,3)));

        set(hRefCur,XData=teb.ReferencePath(:,1),YData=teb.ReferencePath(:,2));

        hTEBPath2_2.Matrix(1:3,:) = [eul2rotm([0 0 curpose(3)],'XYZ')
[curpose(1:2)';0]]

            teb.RobotInformation.Dimension(1),teb.RobotInformation.Dimension(2));

        if needLocalReplan

            continue;

        else

            if needFreeSpaceReplan

                error('Need replan');

            end

        end

        timestamps = tstamps + simtime;

        tVis = 0;

        tPlan = 0;

    end

    adjustedPath = 0;
```

```
    simtime = simtime + tsIntegrator;

    tVis = tVis + tsIntegrator;

    tPlan = tPlan+tsIntegrator;

    velcmd = velocityCommand(velcmds, timestamps, simtime);

       statedot = [velcmd(1)*cos(curpose(3)) ...

                velcmd(1)*sin(curpose(3)) ...

                velcmd(2)];

    curpose = curpose + statedot * tsIntegrator;

    curvel = velcmd;

end
```