Assignment 3
Planetary Rover Planning - Optimal Orbit Transfer

AERO4701 Space Engineering 3

May 7, 2016

In the first part of the assignment you will implement a set of traversability evaluation and motion planning algorithms for a Mars Rover path planning problem. In the second part of the assignment you will implement a nonlinear Program to solve for the optimal time and Delta-V burns that minimize the total change in velocity magnitude for a two-burn orbit transfer from a satellite park orbit into a GEO orbit.

General

- This assignment contributes 40% to your final mark.
- Late submission will result in a 20% (20 marks out of a possible 100) reduction for each day late, starting from 5pm on the day the assignment is due and including weekends.
- Any special consideration requires you to go through the Special Consideration form process. Only major illnesses/misadventures will be considered.
- Minor Plagiarism will attract 0 marks for all those involved for the whole assignment along with the issue being followed up with the T&L Director and then onto the University Registrar. Likewise with major plagiarism except that an automatic failure of the whole course will be sought. No assignment will be marked unless the Student Plagiarism Compliance Sheet is filled out and submitted.
- This assignment should take the average student 40 hours to complete.

Submission

- This assignment is due on Friday 3rd June 2016, at 23:59pm by email to Kelvin Hsu (yhsu9975@uni.sydney.edu.au) **and submission via TurnItIn**. Please submit the report on TurnItIn, and a zip file containing both the code and report through email.
- You will provide one document/report file for the assignment in PDF format named "report.pdf". The report will include the sections "Question 1", "Question 2", and "Appendix" with subsections for each question. Each question will contain the subsections "Introduction", "Methodology", and "Results/Discussion".

- Use plots from your MATLAB code to illustrate your findings. You may use an appendix at the end of your report for your figures and reference them in the body of your report. The page limit **per each section** is 5 pages (not including the appendix). Pages over the limit will not be marked.
- For each question you will submit working MATLAB code. For each question, make sure there is a main file named mainQN.m where N is the question number. This main script will be the ONLY script run by the tutors during marking and this script should run all of your code and produce all of your plots and simulations for the questions.
- All MATLAB code must be commented so that the code can be read and understood without the aid of the report. General code quality and good software engineering practices are part of the marking criteria in this assignment.
- If your code does not work, you will receive 0 marks for the code section of the marking unless you provide a valuable explanation in the report as to why it didn't work.
- Place all of your code for each question and your report in one .zip file. Name the .zip file "SID_Assignment3.zip" where SID is your student number. Email this file to the tutor when submitting your assignment.
- It is your responsibility to check that you have sent all the files for each question. We will not follow you up if we find the code or report missing from any of the questions in the email, and if it is not there, we will assume you did not attempt that question.

Question 1

This question provides an introduction to grid-based robot motion planning. It is your task to implement a series of traversability map generation and motion planning algorithms for the hypothetical Mars Rover, Orville. The Orville rover has a similar design to the Spirit and Opportunity Mars Exploration Rovers, sent to the red planet in the early 2000s. The rover has landed at a relatively rugged site and needs to explore the local terrain for interesting geological features. Your map generation and motion planning algorithms will be used to guide the Orville rover through the terrain during its operations.

To complete this assignment section, download the relevant zip file. Once uncompressed, this folder contains the following files and directories:

- `Assignment3Q1.m` - The main file to be used to run experiment code.
- `data_sets` - A directory containing all of the necessary .mat files containing terrain and obstacle data needed for planning.
- `utility_functions` - A directory containing a collection of functions that perform various housekeeping tasks within `Assignment3Q1.m`.
- `student_functions` - A directory where you will store your functions. You will need to create this directory and submit it as part of your assignment deliverables.
- `p_functions` - A directory in which content-obscured functions are stored. Each of these content-obscured functions returns 'correct' output. You may use these functions to compare against your functions' output.

The following sections contain a description that is equivalent to the description provided in the relevant cells of `Assignement3Q1.m`. Beside each section title is the number of functions that you will have to implement as part of the section along with the number of marks allocated to each section. **Before attempting to solve this assignment, we suggest that you read through this document and the `Assignment3Q1.m` file in their entirety so that you may better understand the expectations of this assignment.**

Cell 1: Martian Terrain Map (0 Functions - 0 Marks)

The terrain that the Orville rover will be operating on is shown in Figure 1. A blue diamond denotes the start location of the rover, while a red star denotes the goal location of the rover. Unlike the Spirit and Opportunity MERs, the Orville rover has the scout Wilbur UAV on board, which is capable of exploring the local terrain around the Orville rover. From the Wilbur UAV's imagery, along with Mars Reconnaissance Orbiter (MRO) imagery, a point cloud of the terrain shown in Figure 1 is generated. This data set is provided in the file `localTerrainPointCloud.mat`. The point cloud will be used to evaluate the traversability of the terrain.

Cell 2: Gridded Map (1 Function – 5 Marks)

Your first task is to define an occupancy grid over which we can plan a path for the rover. Each grid cell is to have a size of 20 cm by 20 cm, roughly the footprint area of a single Orville rover wheel. It is your task to write a function that returns the variable `map`. `map` is a matrix with every element of the matrix representing a single cell in the occupancy grid.

You shall implement the function `generateGrid` with the following interface:

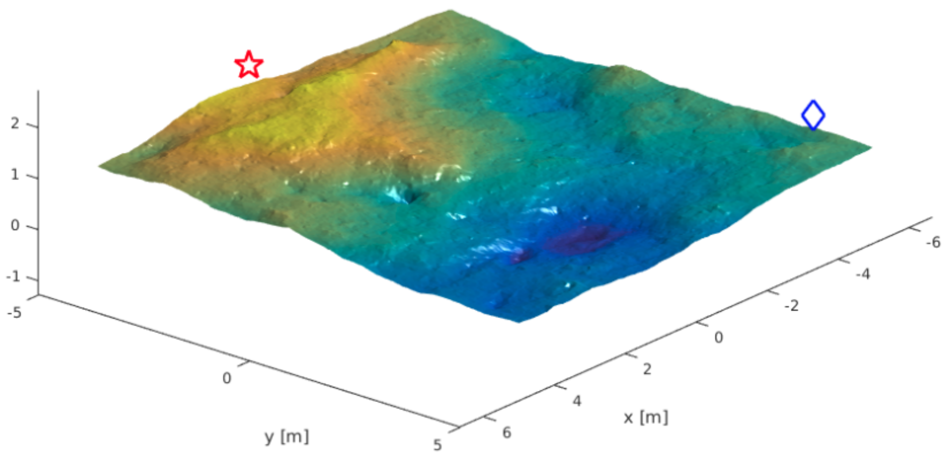


Figure 1: The terrain that the Orville rover will be exploring.

```
map = generateGrid(mapDim,cellDim,startPos,goalPos)
```

that returns a **map** matrix that has the correct size to ensure complete coverage of the 10 m by 10 m terrain and a cell size of 20 cm by 20 cm. The returned map shall have all matrix elements with a value of 1 except for the start and goal cells, which will have values of 2 and 3 respectively. These values correspond to an enumeration summarized in Table 1 that denotes the type of the cell. This enumeration is detailed below and will be used throughout this assignment if not mentioned otherwise.

Enumeration	**Colour**	**RGB Value**	**Cell Type**
1	white	[1,1,1]	clear cell
2	blue	[0,0,1]	start
3	red	[1,0,0]	goal
4	black	[0,0,0]	obstacle
5	cyan	[0,1,1]	on list
6	orange	[1,0.5,1]	visited
7	yellow	[1,0,0]	path

Table 1: Grid cell enumeration for the `map` array.

Cell 3: Configuration Space (1 Function – 5 Marks)

The Orville Rover ground control hazard surveillance team has identified some obvious hazard areas from Wilbur UAV on-board camera data and orbital imagery. They have given the rover mobility planning team (you) a list of coordinates in the local terrain reference frame that located the centre of the hazard areas (obstacles). The `obstacles` matrix contained within `hazards.mat` contains the x,y coordinates and obstacle radii r in each row in meters.

In this section it is your task to take this list of hazard areas and represent them within the map that you generated in the previous section. A hazard area or obstacle is indicated in the map using enumeration code 4 as detailed in Table 1.

You will have to write a function

```
map = generateObstacles(map,obstacles,mapDim,cellDim)
```

that assigns the appropriate map cells as obstacles.

Remember that you are generating the configuration space of the robot and not simply the occupancy grid. You must therefore remember to inflate the obstacles by a radius equivalent to the radius of the rover's footprint (0.5 m).

Cell 4: The GESTALT Algorithm (5 Functions – 20 Marks)

Now that you have a configuration space, you shall generate a traversability map that evaluates the 'free' terrain and gives each cell in the map an associated traversability cost. Each cell will be given a traversability cost between 0 and 255. This score is an estimate of how traversable the associated cell is. The higher the traversability cost, the less traversable the terrain is. The rules for estimating this traversability map score are detailed in the Mars Rover Planning lecture slides.

A point cloud of the local terrain around the Mars Rover is provided in the file `terrainPointCloud.mat`. The GESTALT algorithm iterates over every cell in the map and finds the points within the point cloud that are within a radius equivalent to the footprint of the rover. This operation is performed by the function `getRoverFootprintPoints`. You are required to implement this function. It is suggested that you look into using the `rangesearch` inbuilt MATLAB function to help with this implementation. You may find out more about this function by typing `help rangesearch` into the MATLAB command window. The interface for `getRoverFootprintPoints` is

```
points = getRoverFootprintPoints(pos,roverFootprintRadius)
```

The output `points` is a $N_{patch} \times 3$ array of points within the rover footprint patch, with each column corresponding to the x,y and z coordinates of the N_{patch} points respectively. The input arguments `pos` is a 1×2 vector that describes the x and y position coordinates of the cell, while the argument `roverFootprintRadius` is the radius of the rover's footprint in meters.

The next step in GESTALT is to fit a plane to the patch of points returned by `getRoverFootprintPoints`. This operation will be performed by the function `fitPlane`. The interface for this function is

```
[n,p] = fitPlane(points)
```

where `n` is the unit vector normal to the fitted plane and `p` is 1×3 vector describing an average point of the patch points.

With this plane and the patch points, terrain hazard evaluation may be conducted. The three hazard evaluation functions to be implemented evaluate the incline and roughness of the terrain along with any step obstacles that may be present. To evaluate terrain incline, the function `pitchHazardEval` finds the angle between the fitted plane's normal vector and the xy plane. To evaluate roughness, the function `roughnessHazardEval` finds the standard deviation of the residuals of the plane fit. To evaluate step obstacles, the function `stepHazardEval` finds the maximum height difference between points in a cell patch. Each of these hazard evaluation functions returns a cell score.

The interfaces for each of these functions are:

```
scoreStep = stepHazardEval(points,roverClearanceHeight)
```

```
scorePitch = pitchHazardEval(n,roverMaxPitch)
```

```
    scoreRoughness = roughnessHazardEval(points,n,p,roverClearanceHeight)
```

The total traversability score of a cell is taken as the maximum of the incline, roughness and step scores.

Cell 5: Dijkstra's Algorithm (2 Functions – 15 Marks)

Now that you have a traversability map, you shall plan a path for the Orville Rover between its designated start and goal positions. You will first implement Dijkstra's algorithm to perform a breadth first search over the traversability map and return a minimum cost path between the start and goal nodes.

The skeleton of Dijkstra's algorithm has been provided for you, however you will have to implement the core parts of the algorithm. First, you will write the function `getNeighbours` that returns the cell indices that neighbour the input cell. The interface for this function is:

```
[neighbours,neighbourInd] = ..
getNeighbours(i,j,map,mapHeight,mapWidth,connectedness)
```

The inputs `i` and `j` are the subscript indices of the current cell, `map` is the traversability map and `connectedness` indicates whether `getNeighbours` returns the 4-connected neighbours or the 8-connected neighbours. As shown in Figure 2, the 4-connected neighbours are the four cells that are directly above, to the left, below and to the right of the current cell. The 8-connected neighbours are all of the 4-connected neighbours in addition to all of the cells that are diagnonally adjacent to the current cell. The outputs of this function include `neighbours`, which is a vector containing the linear indices of the neighbour nodes and `neighbourInd`, a vector containing neighbour indices relative to the current cell. For example, in a four-connected set of neighbours, a neighbouring node directly above the current cell may be marked as 1, while the cell to the left is marked as 2, the cell to the bottom is 3 and the cell to the right is 4. If neighbour 1 were not valid, the returned `neighbourInd` would be [2, 3, 4]. These relative indices may be defined arbitrarily and they make it easier to know which neighbours to consider within the Dijkstra and A* algorithms. If you don't want to use `neighbourInd` simply return it as an empty vector.

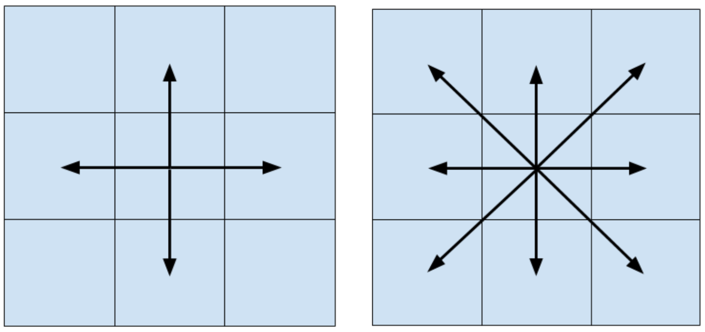
The second function to implement in this section is the body of Dijkstra's algorithm, aptly named `dijkstraBody`. The interface for this function is:

```
[distanceFromStart,parent,open,map] = ...
dijkstraBody(currentNode,distanceFromStart,parent,H,map,mapTraversability,...
neighbours,neighbourInd)
```

where `neighbours` is a vector containing the indices of the neihbour nodes, `distanceFromStart` is an array that contains the current costs to nodes from the start node, `parent` is an array that contains indices of the parent nodes and `H` is an array that contains a `1` for each node that is in the open set and an `Inf` for each node that is not. `currentNode` is the linear index of the current cell that the planner is considering.

Cell 6: A* Algorithm (1 Function – 10 Marks)

The main differentiating component of A* from Dijkstra that causes A* to be a greedy search as opposed to a breadth first search algorithm is the heuristic function. You may choose the type of heuristic to use in this section. It is suggested that you try the Manhattan distance



(a) The four-connected neighbouring cells. (b) The eight-connected neighbouring cells.

Figure 2: The two types of neighbour sets that getNeighbours should be able to return.

or the Euclidean distance heuristics initially. If you want to experiment a bit more, research different heuristics, implement them and describe them in your report. The Manhattan and and Euclidean distance heuristics have been implemented in the provided code for you.
The function to implement in this section is the body of the A* algorithm, `aStarBody`:

```
[g,f,parent,H,map] =...
aStarBody(currentNode,g,f,parent,H,heuristic,map,mapTraversability,...
neighbours,neighbourInd)
```

`g` is the cost-to-come value for each cell in the map, `heuristic` is the guessed cost to-go-values and `f` is the sum of cost-to-come and cost-to-go values.

Cell 7: Plotting the Path (0 Functions – 0 Marks)

The last cell in `Assignment3Q1.m` simply plots the path generated by either one of your planners onto the terrain so that you may visualize the planned path over the terrain.

Extension – Geological Survey Expedition (5 Marks)

Now that you have implemented a traversability map generator and basic path planners for the Orville rover, it is time to put them to use in planning a geological survey expedition. The file `sitesOfInterest.mat` provides five separate sites of geological interest that are to be visited by the Orville rover. Your final task is to use the implemented planners to find a set of efficient and traversable paths between each of these sites of interest. In your report you should present the selected set of paths on the terrain in a figure and you should discuss the methodology you used to find this set of paths and why it is the most efficient and traversable set of paths.

Deliverables

You are required to submit the version of `Assignment3Q1.m` that you used to complete the assignment along with the functions that you have been asked to implement in the section above. Along with this MATLAB code, you will need to submit a report that presents the output of each function as well as a discussion of your understanding of each section and how you implemented the relevant algorithms. If you are unable to implement a function you are allowed to use the relevant p_function in its place, however please indicate that you have done so in both your `Assignement3Q1.m` file as a comment and in your report. Failure to do so will result in a overall assignment mark penalty. If you use a p_function in stead of your own function, you will not be awarded any marks for the code implementation of the relevant section. You will however still be able to gain marks from your discussion of the relevant section.

The section of your assignment report for Question 1 should be no longer than five pages (approximately one page for each assessed cell). This is a MAXIMUM limit. You may be able to provide an appropriate discussion for a specific cell in much less than a page. Figures should be provided in an appendix section that is not included in the page limit.
If you have any questions or notice any mistakes in the assignment, please feel free to contact Will, Wolfram, Steve or Kelvin.

Question 2

In this question you will implement a nonlinear program to solve for the optimal time and ΔV burns that minimize the total fuel usage for a two-burn orbit transfer from a satellite park orbit into a geostationary target orbit.

- The following table specifies the parameters for the satellite park orbit:

Orbital Parameter	**Value**
Semi-major Axis, a (m)	6655937
Eccentricity, e	0
Inclination Angle, i (deg)	-28.5
RAAN, Ω (deg)	0
Argument of Perigee, ω (deg)	0
Mean Anomaly, M_0 (deg)	0
Epoch, t_0 (s)	0

Table 2: Satellite Park Orbit Parameters

- The target GEO orbit is to have a zero inclination angle ($i_{GEO} = 0$). There are no fixed timing requirements (i.e. $t_{0,GEO}$, $M_{0,GEO}$ are free).
- Optimisation parameters are (ΔE_1, ΔV_1, θ_1, ψ_1, ΔE_2, ΔV_2, θ_2, ψ_2) where ΔE is the change in eccentric anomaly, ΔV is the magnitude of change in velocity, and θ and ψ are the elevation and azimuth angles of ΔV burn respectively.
- Determine the radius and velocity required for a GEO orbit.
- Derive the problem equivalence between fuel minimisation and total ΔV minimisation.
- Formulate the optimisation problem mathematically.
- Discuss the dynamical model used to propagate the satellite trajectory. Discuss its role in the optimisation program.
- Discuss how Sequential Quadratic Programming works in the context of fuel minimisation for a two-burn orbit transfer. Identify the five main parts of the SQP algorithm, of which four are optional and not strictly necessary. Briefly discuss the role of each part. Back your discussion with pseudocode or a flow chart (or both).
- Implement a nonlinear program to solve for the optimal parameters that minimise the fuel burn and thus the total change in velocity. **Efficiency, simplicity, clarity, and modularity will be a major part of the marking criteria**.
- Provide ECI plots of your park orbit, transfer orbit and final GEO orbit. Show the change in velocity vector applied at each burn in your plot. Compare the ECI plots of your initial and final trajectories.
- Plot the following quantities against the SQP iterations of the optimisation process.
  - Step Size
  - Error
  - Objective
  - Lagrangian

  - Optimisation parameters
  - Constraints (on the same plot)

- Provide a CLI or GUI to include the use of the following techniques for the optimisation process. Note that some techniques may only make sense in the presence of another technique. You may also choose to research into other optimisation techniques and implement them for complementary marks.
  - BFGS Hessian Update
  - Line Search
  - Merit Function
  - Penalty Parameter
- Discuss how you would estimate the initial values of the optimisation parameters.
- Tabulate your optimal solutions.
- Vary your initial guess for the optimial trajectory from a very good one to a very bad one. Discuss how well your optimiser performs under different initial scenarios.
- Discuss how you would approach the problem if you were given any non-zero inclination for the target GEO orbit. Implement your strategy in your optimiser. Does it work for any angle? How well does it work?
- Discuss how you would approach the problem if you were given any non-zero right ascension of the ascending node for the target GEO orbit. Implement your strategy in your optimiser. Does it work for any angle? How well does it work?
- Write a report to describe the methodology of your optimiser. Use a block diagram or pseudocode to aid your explanation. This report has a 4 page limit (not including appendix).