AAS 21-642

A PRELIMINARY EXPLORATION OF PATH PLANNING FOR INITIAL GUESS CONSTRUCTION IN MULTI-BODY SYSTEMS

Kristen L. Bruchko[*] and Natasha Bosanac[†]

Constructing trajectories in multi-body systems currently relies heavily on a human analyst and, in scenarios may become time-consuming and challenging. Path planning techniques offer one approach to summarizing the solution space and autonomously generating trajectories. In this paper, roadmap generation is used to summarize the solution space. Then, a graph search algorithm is used to construct an initial guess for a transfer. This approach is explored in the context of transfers between L_1 and L_2 Lyapunov orbits in the Earth-Moon circular restricted three-body problem.

INTRODUCTION

Cislunar and deep space exploration are enabled by the successful design of trajectories in the chaotic regimes of multi-body systems. However, constructing a trajectory in these complex environments heavily relies on a human-in-the-loop. Typically, constructing a trajectory that adheres to multiple design objectives requires an adequate preliminary solution and sufficient knowledge of the solution space. This time-consuming process may need to be repeated when mission requirements and constraints change. Current state of the art approaches limit the rapid design of efficient solutions, particularly during the beginning stages of mission concept development. In these scenarios, path planning techniques may offer an alternative approach that reduces the dependence on a human-in-the-loop during initial trajectory construction.

Path planning is a method of computing collision free paths subject to constraints or dynamics in environments containing obstacles.[1] Specifically, probabilistic path planning has been used in robotics, artificial intelligence, and control theory to efficiently produce solutions in complex environments.[2] Roadmap generation, one common method of probabilistic path planning, utilizes incremental sampling to summarize the environment and a search algorithm to find valid paths from a start configuration(s) to a goal configuration(s). The first step in roadmap generation is to construct the roadmap, which is a map of the solution space. Configurations sampled from the solution space are added to the map as nodes and local paths connecting neighboring nodes are added to the map as edges. The roadmap is constructed to capture the global dynamics of the design space, including sensitive regions, such that path planning solutions may be computed efficiently. Roadmap generation has been demonstrated to solve problems in high-dimensional spaces with probabilistic completeness, i.e., the probability of finding a solution if one exists approaches unity as the number of nodes increases.[3]

[*]Graduate Research Assistant, Colorado Center for Astrodynamics Research, Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO, 80303.

[†]Assistant Professor, Colorado Center for Astrodynamics Research, Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO, 80303.

The challenges of trajectory design are similar to the challenges addressed in path planning problems. In astrodynamics, Das-Stuart, Howell, and Folta implemented automated path finding techniques to recover an initial guess in a variety of transfer scenarios using Dijkstra's graph search method and a database of fundamental solutions.[4] In addition, a tree-based search has been used by Starek et. al. to construct propellant-optimized solutions for rendezvous and proximity operations near circular orbits in a dynamical environment governed by the modeled by Clohessy-Wiltshire-Hill equations.[5] Trumbauer and Villac have also used a precomputed database of periodic orbits and invariant manifolds to generate a directed graph and design transfers between periodic orbits in lower-fidelity models using discrete search algorithms, which are then corrected and optimized during onboard trajectory redesign.[6] Tsirogiannis has used a graph based method to construct trajectories using segments along precomputed periodic orbits.[7] These contributions motivate further analysis of the utility of roadmap generation to summarize the solution space in multi-body systems, with the goal of constructing initial guesses for trajectories in complex environments efficiently and without heavy reliance on a human-in-the-loop.

To construct a useful roadmap in a multi-body system, a general understanding of the environment is valuable. To reduce the complexity of the design process, trajectory designers often begin by designing a guess in simplified models. One well-known approximation is the Circular Restricted Three Body Problem (CR3BP), which is a useful model for a spacecraft near two larger celestial bodies, such as the Earth and Moon. In this approximate model, dynamical structures exist such as periodic orbits and invariant manifolds, to govern natural transport within the system. These structures are often directly computed via dynamical systems theory (DST).

In this paper, a preliminary roadmap is generated in the Earth-Moon CR3BP in the vicinity of the L_1 and L_2 libration points using DST to summarize part of the solution space, expanding the procedure presented by Trumbauer and Villac.[6] After the roadmap is generated, a global search algorithm is used to construct initial guess trajectories between an L_1 periodic orbit and an L_2 periodic orbit. To summarize the solution space of possible trajectories for a spacecraft in the CR3BP, states along periodic orbits and their associated invariant manifolds are sampled to form the nodes of the roadmap. A description of the method used for sampling the nodes to construct an efficient representation of the solution space is presented. By using DST to bias the sampling to a restricted subset of the solution space, the goal is to create a useful and efficient roadmap that will support transfer design. Edges are added to the roadmap based on a selected distance metric to determine neighboring nodes. The edges will then be weighted to reflect continuity and accessibility, using metrics such as geometry, state discrepancies, and energy levels.

Once a sufficient roadmap is generated, the procedure for constructing an initial guess trajectory is demonstrated. The time the planner spends generating a sufficient roadmap is about 95% of the total preprocessing time.[8] Multiple queries may then be made to construct initial guess trajectories for a variety of initial and final configurations or paths. To demonstrate the applicability of utilizing roadmap generation for trajectory design, initial guesses for transfers between an L_1 periodic orbit and an L_2 periodic orbit in the Earth-Moon CR3BP are constructed using the foundational Dijkstra's search algorithm, a modification of the A* algorithm. The generated initial guess trajectories are then corrected via multiple shooting to enforce continuity and energy level constraints. Successful demonstration in a preliminary example supports further advancements of the roadmap generation process and demonstration in a wider variety of transfer design scenarios such as spatial or low-thrust enabled motion in multi-body systems, with the ultimate goal of reducing the burden on the trajectory designer.

BACKGROUND: DYNAMICAL MODEL

The Circular Restricted Three Body Problem (CR3BP) is a nonlinear dynamical model that is used to approximate the motion of a spacecraft operating in a multi-body system. The spacecraft is assumed to possess negligible mass and is influenced by the gravity of two primary bodies, such as the Earth and the Moon. The larger body, P_1, and the smaller body, P_2, both are assumed to be constant point masses and travel in circular orbits about their mutual barycenter such that the center of mass lies on the $P_1 - P_2$ line. Then, mass, time, and distance are nondimensionalized via characteristic quantities m^*, t^*, and l^*. Specifically, m^* is set equal to the total mass of the system, t^* is defined such that the mean motion of the primaries is unity, and l^* is set equal to the assumed constant distance between the primaries. Using these assumptions, an orthogonal reference frame that rotates at the same rate at the primaries is constructed, removing the time dependence from the system. The $\hat{x}$ axis is directed from P_1 to P_2, the $\hat{z}$ axis is defined in the direction of the primary system's orbital angular momentum, and the $\hat{y}$ axis completes the right-handed triad. As a result, P_1 and P_2 appear stationary in the rotating frame. The state of the spacecraft is then written in the rotating frame with respect to the system barycenter as $\boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$. The nondimensional equations of motion for a spacecraft in the CR3BP are then written as:

$$\begin{aligned}
\ddot{x} &= 2\dot{y} + x - \frac{(1-\mu)(x+\mu)}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3} \\
\ddot{y} &= -2\dot{x} + y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} \\
\ddot{z} &= -\frac{(1-\mu)z}{r_1^3} - \frac{\mu z}{r_2^3}
\end{aligned} \tag{1}$$

where $r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$ and $r_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2}$ are the distances of the spacecraft from P_1 and P_2, respectively, and $\mu = M_2/(M_1 + M_2)$ is the mass ratio of the system. One integral of motion, commonly called the Jacobi constant, is defined as:

$$C_J = (x^2 + y^2) + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} - \dot{x}^2 - \dot{y}^2 - \dot{z}^2 \tag{2}$$

This energy-like constant supplies insight into the motion of a spacecraft and the accessible regions of the system.

At a single value of the Jacobi constant, the CR3BP admits a variety of natural dynamical structures such as equilibrium points, periodic orbits, and invariant manifolds.[9] Within the CR3BP, there are five equilibrium points that lie in the motion of the primaries: three collinear points that lie along the $\hat{x}$ axis, commonly denoted L_1, L_2, and L_3, and two triangular points that form an equilateral triangle with the primaries, commonly denoted L_4 and L_5. Continuous families of periodic orbits also exist throughout the system, including near each of the equilibrium points.[9] As an example, a subset of the L_1 Lyapunov periodic orbit family is depicted in Figure 1. Unstable periodic orbits possess stable and unstable invariant manifolds, natural motion that approaches and departs the periodic orbit, respectively, as time tends to infinity. These fundamental solutions have been observed to govern natural transport in the CR3BP and are crucial in trajectory design. However, analysis may become challenging in some scenarios due to the complexity of the solution space. Since these dynamical structures cannot be computed analytically, they are traditionally computed numerically via a combination of dynamical systems techniques and a corrections algorithm.

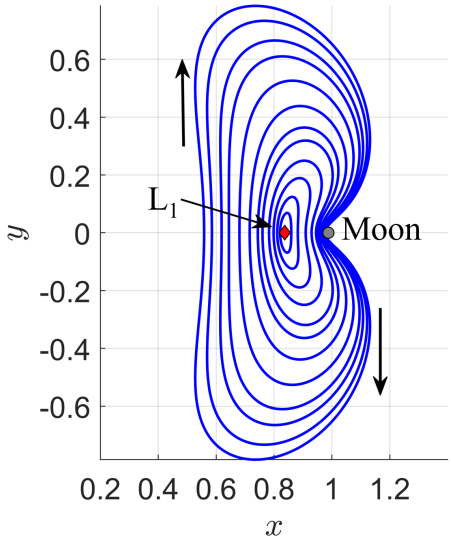

Figure 1. A subset of the L_1 Lyapunov periodic orbit family, displayed in the rotating frame.

BACKGROUND: NUMERICALLY CORRECTING TRAJECTORIES

To compute transfers in the CR3BP from a discontinuous initial guess, a multiple-shooting corrections algorithm is used to solve a two-point boundary value problem. First, an initial guess is discretized into multiple arcs. Nodes are located at the beginning of each arc and at the last state along the final arc. This decomposes the initial guess into n nodes with $n-1$ integration times, assuming each arc has an independent integration time. The state vector at each node and integration time along each arc are used to define a $(7n-1)$-dimensional free variable vector, $\bar{\boldsymbol{X}}$:

$$\bar{\boldsymbol{X}} = [\bar{x}_1, \bar{x}_2 \ldots, \bar{x}_n, t_1, t_2, \ldots, t_{n-1}]^T \tag{3}$$

To recover a continuous trajectory, state continuity constraints are enforced between neighboring arcs. These constraints form a $(6n-6)$-dimensional constraint vector $\bar{\boldsymbol{F}}$:

$$\bar{\boldsymbol{F}} = [\bar{x}(t_1) - \bar{x}_2, \ldots, \bar{x}(t_{n-1}) - \bar{x}_n]^T \tag{4}$$

Using these definitions, the free variable vector is updated until the norm of the constraint vector equals zero within a desired tolerance. This is accomplished by using Newton's method. The update equation at the i-th iteration is equal to:

$$\bar{\boldsymbol{X}}_{i+1} = \bar{\boldsymbol{X}}_i - D\bar{F}(\bar{\boldsymbol{X}}_i)^T (D\bar{F}(\bar{\boldsymbol{X}}_i) \cdot D\bar{F}(\bar{\boldsymbol{X}}_i)^T)^{-1} \bar{\boldsymbol{F}}(\bar{\boldsymbol{X}}_i) \tag{5}$$

where $D\bar{F}(\bar{\boldsymbol{X}})$ is the Jacobian matrix, calculated analytically for the formulation of multiple shooting used in this paper. Because the Jacobian is rectangular and, therefore, not invertible, the minimum norm solution is used in the update equation.

BACKGROUND: PATH PLANNING

Path planning algorithms are commonly used in robotics, artificial intelligence, and control theory, to autonomously explore a solution space and plan a sequence of collision free paths from a

start configuration to a goal configuration while respecting all environment and robot constraints.[2] Originally, path planning was commonly used in robotics for environments with continuous spaces, such as the common piano mover problem of moving a piano from one room to another without hitting anything.[2] In recent studies, these autonomous algorithms have been modified to handle more challenging problems, such as planning with uncertainty and differential constraints. Initial problems included robotic arms in assembly lines, where the motion of the robot was limited and linear. Current problems being addressed include rovers for planetary exploration that compute the environment uncertainty with sensors or cars that travel autonomously on highways that are required to follow the cars dynamics.[10] A variety of algorithms exist in path planning, each having many adaptations, designed to be applicable to one general problem: how to specify the motion of a robot autonomously within an environment.[3] Common path planning algorithms are typically categorized as potential functions, cell decompositions, or sampling-based algorithms.[3]

Probabilistic Roadmaps

A probabilistic roadmap (PRM) is a sampling-based path planning algorithm that generates a roadmap of the environment. The roadmap is a graph, $G = (N, E)$, that summarizes the solution space with nodes, N, and edges, E, while adhering to any environment constraints, dynamical equations, or obstacles. An advantage for PRM over other graph methods is the roadmap can be searched for multiple query configurations without the need to reconstruct the roadmap. Once the solution space is summarized, multiple paths can be found for a variety of start and goal configurations. Two important characteristics of this sampling-based planner are that it is efficient at planning in high-dimensional spaces with many different constraints and it may potentially achieve probabilistic completeness.[3] Probabilistically complete planners guarantee that if a path exists for a given roadmap, the planner will eventually find it.

Constructing and searching a PRM consists of two main phases: the learning phase and the query phase. The learning phase focuses on creating a roadmap of the solution space by sampling nodes and using a local planner to connect neighboring nodes to form the edges. The nodes, $n \subset N$, represent valid configurations in the solution space. Configurations represent a complete specification of the location of every point on the robot being planned for, such as state representations for translational and rotational components.[11] The definition of a valid configuration depends on the environment. Example definitions include configurations that do not intersect any obstacles that may be present or configurations that adhere to any dynamical equations. Configurations are sampled until the solution space has been sufficiently summarized and all nodes in the roadmap are considered valid. Originally this planner was based on uniform sampling of the solution space to construct the nodes of the roadmap. The uniform sampling implementation is called basic PRM. Recent adaptations of the PRM algorithm demonstrate that other sampling schemes, such as sampling more in difficult or sensitive areas based on the density of obstacles in a region, or discretizing the solution space into a grid and sampling along the grid lines, are useful and efficient depending on the solution space.[3] Once the nodes are added to the roadmap, a local planner is then used to connect neighboring nodes to each other to form the edges, $e \subset E$, of the roadmap. Edges represent collision-free, valid paths between nodes and may be constructed with a variety of methods. A node is connected to its k-nearest neighbors, which may be defined by a distance metric (e.g. the euclidean distance) or if the environment is simple, every other node in the roadmap. Once the roadmap is considered complete, i.e. all nodes and edges have been added, the query phase uses a global planner to construct a path from user-defined start configuration(s) to goal configuration(s) by connecting the edges of the roadmap together. The global planner will search the roadmap based

on a graph search method or algorithm. A conceptual example of the key steps in PRM are depicted in Figure 2. In this example, black squares represent static obstacles, grey circles represent sampled nodes, and dashed lines represent edges. The goal is to find a valid path from the start node (blue circle) to the goal node (red circle). Nodes are sampled first, discarding any nodes that are invalid, i.e. nodes that intersect with the obstacles. Next, the local planner connects nodes to its k-nearest neighbors, where $k = 2$ in this example. Finally, a global planner searches for a path from the start node to the goal node, highlighted in a solid black line.

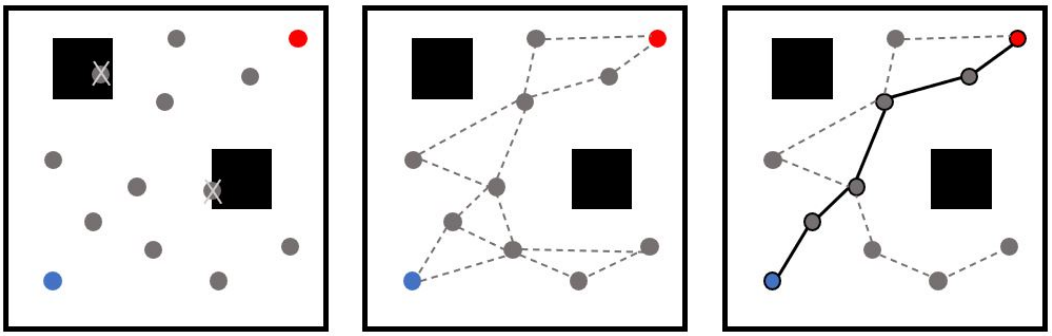

Figure 2. Example of the probabilistic roadmap planning algorithm: a) sample nodes, b) connect nodes to create edges, and c) search the completed roadmap for a path.

Roadmaps may be constructed using undirected or directed graphs, based on the possible orientation of the nodes in an edge pair. Directed graphs have edges that may only be traversed in one direction: from one initial node to its neighbor along the connecting edge. Undirected graphs have edges that may be traversed in either direction: the initial node may be either node in the edge pair. Graphs may also be constructed as weighted graphs, $G = (N, E, W)$, where each edge has an associated weight to it. Weighted graphs may help the selected search algorithm find a path with desired characteristics, since edge weights are suited to the environment.

Graph Search

Discrete search algorithms are used to search the roadmap summarizing the solution space for possible paths. A wide variety of discrete search algorithms, each searching the solution space uniquely, are suited to different environments. Search algorithms are assessed for completeness, whether or not the algorithm will terminate with a correct result, and their computational efficiency. An efficient search minimizes the total number of nodes visited, improving computationally efficiency because it does not necessarily search the entire space and visit every node. A complete search algorithm will always terminate in finite time, either providing a solution path if one exists or determining one does not exist. An incomplete search algorithm will continue searching for infinite time, such as searching an unbounded solution space.

There are three main categories of search algorithms: breadth first searches, depth first searches, and best first searches. Each of these algorithms generates a search tree, but explores the path options differently. Breadth first searches explore all children of the root node first, then explores all of the grandchildren, and repeats until a desired node is found or the graph was entirely searched. A child node is a node connected to the parent node through an edge, whereas a grandchild node is connected to the parent node through two edges, and so on for the remaining levels of children. Breadth first searches explore vertically first, then moving through the graph left to right. This type

of search algorithm produces the shortest path in terms of the number of edges linked together, but it does not guarantee an efficient search. Beginning again at the root of the tree, depth first searches select one parent node, continuously searching through that node's children, until all children are explored or the desired goal node is reached. When all children have been explored, the algorithm returns to the parent node and repeats the process of searching through the next set of children along the next parent node. Depth first searches are typically well-suited for scenarios where it is important to search the entire space. Depth first searches explore horizontally first, then moving through the graph top to bottom. In environments where it is not important to search the entire graph, but instead produce the best path according to a predefined objective, best first search algorithms are useful. Best first searches, also called informed searches, explore the graph efficiently using predefined environment knowledge. In this paper, best first search algorithms are used.

The A* algorithm is a best first search algorithm that explores a graph by prioritizing the order of nodes searched based on which nodes are the most promising. Promising nodes have a lower cost associated with them. Rather than depth or breadth first, this cost is how A* searches through a graph. The cost of a node is calculated using two components: the cost of the path from the start to the current node, $g(n)$, such as the total distance traveled or total time traveled, and the estimated cost to the goal from the current node, $h(n)$ labeled the heuristic cost, such as the estimated distance from a current node to the goal node. These two components of the cost function estimate the total cost of a path from start to goal. The current cost of a path, $g(n)$, is known for each node, because the algorithm already knows where it searched prior to that node and the path required to take to arrive at the current node. The heuristic cost of a path, $h(n)$, is estimated for each node, because the algorithm does not know anything about what nodes lie between the current node and the goal node. Therefore, an optimal heuristic will guide the search algorithm to the goal node faster. Then, the total cost function is the sum of the each component, defined as:

$$f(n) = g(n) + h(n) \tag{6}$$

where $f(n)$ is the total cost of the path for a specific node. A* algorithm will search for a path that minimizes $f(n)$. To efficiently search the graph, the algorithm generates a search tree of the graph and searches based on two lists: a priority list and a closed list. The priority list contains the list of nodes to search in the current iteration. The cost of each node, $f(n)$, determines the order of the priority list and the order of nodes being searched. Nodes with the lowest cost will be first in the priority list, therefore will be searched first. Along with the nodes, the priority list stores each node's parent and the total cost of traversing from the selected start node to the node in the list. At each iteration in the search algorithm, the priority list is sorted based on the total cost of each node. The closed list contains the nodes that have already been searched. Maintaining a closed list guarantees the algorithm does not search the same node twice, supporting efficiency. An outline of the A* algorithm is briefly described as follows:

1) To begin, add the start node(s) to the priority list
2) Remove the start node with the lowest initial cost from the priority list, add it to the closed list, and add all of its neighboring nodes to the priority list
3) Sort the priority list by the total cost for each node in the list
4) Remove the best node with the lowest total cost and add all neighboring nodes that do not already exist in the priority list
5) Sort and repeat until the goal node(s) is reached or all nodes have been searched and are contained within the closed list

The A* algorithm will find an optimal path if its heuristic is also admissible, by always estimating the cost to the goal that is less than or equal to the actual cost to the goal within the graph. This means the heuristic should never overestimate the cost to goal. For example, if the only valid path from a start node to a goal node was to traverse around an obstacle that lies between them, the heuristic estimate from the start to goal could be along a straight line drawn between the nodes. This heuristic is not valid because the path would intersect the obstacle, but it is only an estimate to guide the algorithm in the direction of the goal. The heuristic should be designed such that $h(goal) = 0$. If the heuristic is designed poorly and is not admissible, A* still terminates, but the search will be inefficient. A* always terminates, providing the solution path if one exists or terminating if one does not exist. For that reason it is considered a complete planner. The proof of optimality and completeness is described in greater detail by Stuart J. Russell and Peter Norvig.[12]

Dijkstra's algorithm is an adaptation of the A* algorithm, where the heuristic cost of each node is $h(n) = 0$. Thus, the total cost of the path is only calculated using the known cost from the start in the graph. Dijkstra's algorithm explores the nodes with the lowest cost from the start, therefore producing the path with the lowest overall cost. If A* is adapted such that the cost from the start does not matter and $g(n) = 0$, the search becomes the greedy implementation of A*. This greedy A* only searches based on the estimated heuristic function of each node and what the algorithm believes is the best path to goal, but may not produce the lowest cost path.

TECHNICAL APPROACH

In this paper, PRM is explored in a simplified trajectory design scenario by summarizing the solution space and constructing trajectories. To construct an initial guess for a transfer between two orbits, a portion of the solution space is summarized via a roadmap. The roadmap is searched to produce a discontinuous initial guess. In addition to the learning and query phases typically used in roadmap generation, a preprocessing phase is used to represent the desired solution space within the CR3BP using fundamental solutions such as periodic orbits and their associated invariant manifolds. Precomputing a database of fundamental solutions to serve as a basis for the roadmap has been demonstrated to summarize the solution space efficiently.[4–6] This approach is summarized in Figure 3. In this analysis, the parameters required to construct the roadmap are adapted to the CR3BP and each phase of the process is outlined in the remainder of this section. Specifically, this preliminary exploration of applying PRM to the CR3BP focuses on constructing initial guesses for transfers between L_1 and L_2 Lyapunov orbits in the Earth-Moon CR3BP.

Phase 1: Computation of Key Dynamical Structures

The first phase focuses on the computation of natural dynamical structures that govern the solution space in the CR3BP. In this simplified scenario, these structures include a desired initial and final periodic orbit, as well as trajectories along their associated invariant manifolds. Of course, as the roadmap is expanded in ongoing work, a wider variety of dynamical structures must be computed. Furthermore, this paper limits the exploration of this analysis to only natural transfers, but may be adapted to allow for low-thrust maneuvers to be performed along the transfer. The selected fundamental solutions are then discretized and used in the learning phase to bias the sampling process. Computing these structures for the roadmap a priori ensures the construction of the graph adheres to the dynamics of the CR3BP. States along the trajectories are then used as a data subset to bias the construction of the roadmap to the region of interest. While periodic orbits and natural trajectories are continuous solutions in the CR3BP, the discretization of these structures requires

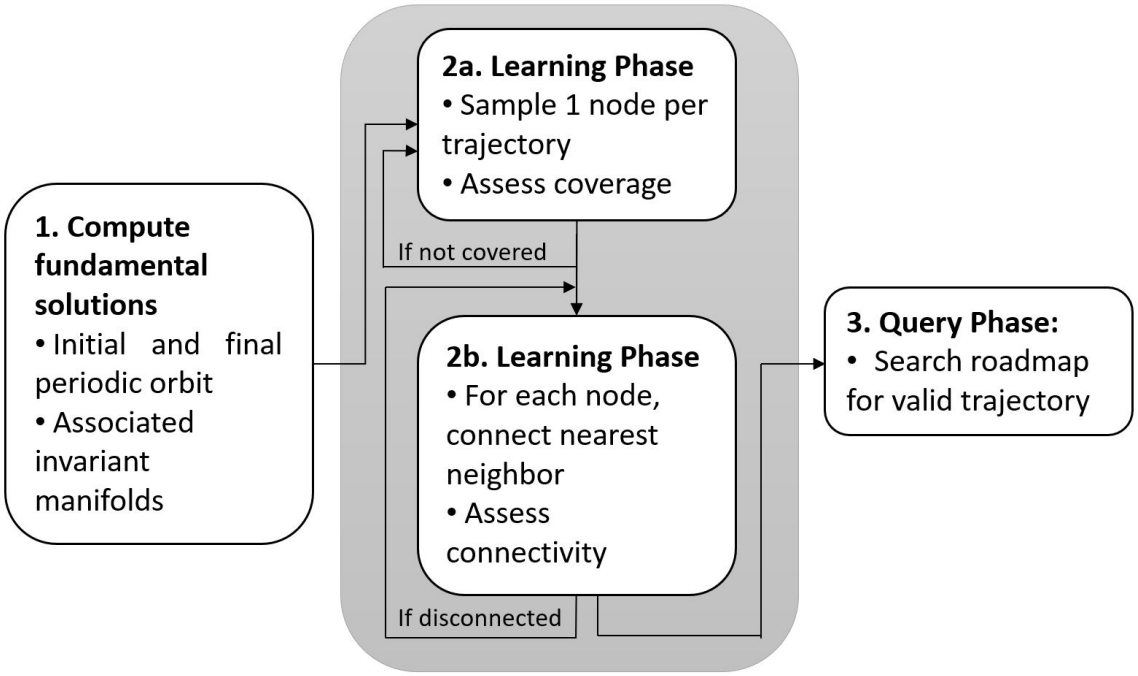

Figure 3. Overview of procedure used to construct an initial guess for a transfer between an L_1 and L_2 Lyapunov orbit in the CR3BP using roadmap generation in this paper.

a trade off of available computational resources. Although a finer discretization supports a more thorough exploration of the solution space, it is also more computationally intensive and requires a larger amount of memory storage.

Phase 2: Learning Phase

During the learning phase, a roadmap of the solution space is constructed. The roadmap is a weighted, directed graph, $G = (N, E, W)$ containing nodes and edges that is a summarized representation of the information received from the previous phase. A node, $n \subset N$, is represented by a randomly sampled state in the solution space. An edge, $e \subset E$, is a connection between two neighboring nodes, $(n_1, n_2) \subset N$, in the roadmap. To efficiently explore the solution space to construct an initial guess for a transfer, the methods to construct the nodes and edges must be adapted to the CR3BP. As described in Figure 3, the roadmap is constructed through an iterative process of sampling nodes from the precomputed fundamental solutions until the portion of the solution space is deemed covered. Then, the connection of nodes to form the edges is also iterated until the roadmap is connected.

Although the selected sampling scheme may vary depending on the environment, a random uniform sampling scheme is implemented as in basic PRM. To ensure all nodes adhere to the dynamics in the CR3BP, nodes are specified using a full state vector as well as a propagation time sampled from the data subset. Nodes are added one trajectory at a time along each fundamental solution to produce a even distribution of nodes across the precomputed trajectories, yet still randomly covering the solution space along each individual trajectory. For each trajectory, the arc length is computed

through integration along with the equations of motion. The arc length is calculated from the initial state on the trajectory, the first step off the periodic orbit, to the final state where each trajectory intersected the Moon hyperplane in the positive direction. A random arc length value is sampled from the bounds of the trajectory's computed arc length. The state along the trajectory that corresponds to the value of the arc length sampled serves as the initial state of the node. By sampling along the arc length, the nodes are probabilistically equally distributed along the trajectory. The initial state is then propagated until the node's arc length is a fixed length, determined by the maximum arc length in Table 1. The propagation time required to reach the fixed arc length is then saved along with the initial state, completing specification of the node. Nodes are continuously added to the roadmap along each trajectory until the solution space is sufficiently summarized. The roadmap is considered a summarized representation of the solution space when the nodes cover the same area in position space as the input trajectories.[13] To determine if the nodes cover the same area, two calculations are performed. Initially, the total area of all precomputed trajectories is calculated. As nodes are added to the roadmap, each node covers a region defined by its neighborhood. The neighborhood is defined as a hypersphere around the initial state with a radius equal to one-half the maximum arc length. Then, the total area the nodes cover is determined by the union of the neighborhoods covered by each node. Nodes are added incrementally to the roadmap until the nodes cover the total solution space. An example of the coverage of the roadmap is displayed in Figure 4. The area defined by the precomputed trajectories is displayed in light blue. For each node sampled, the initial state is shown as a black circle. Each node covers the area around its initial state, shown as the grey circles. This example roadmap is not covered at this iteration because the nodes only cover 93% of the area, thus more nodes are required.

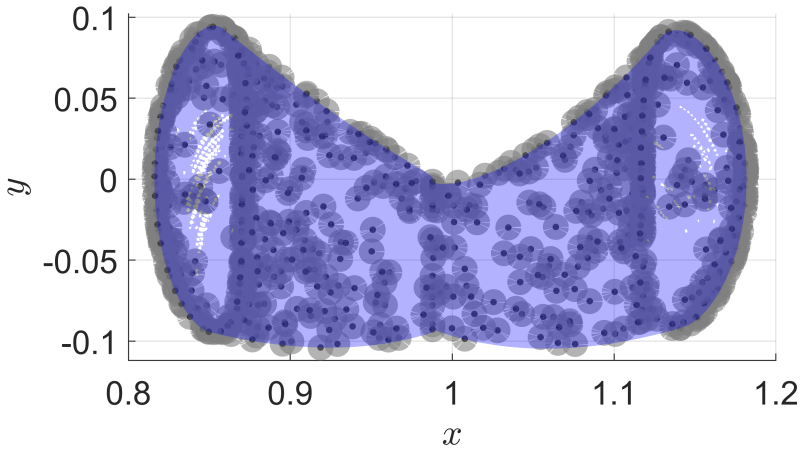

Figure 4. Coverage of the roadmap defined by its nodes.

Once the roadmap is fully covered, edges are added. Edges are the node connections based on the selected neighboring node strategy employed, but do not have a physical representation in the CR3BP. These connections represent the discontinuities between the end of the arc described by a

node and its neighbors. For each node, called the parent node, the nearest neighbor is identified that does not already form an edge to the parent node. If the distance between the final state along the arc associated with the parent node and the initial state along the arc associated with the neighboring node is within a predefined distance, it is saved as an edge. Each valid edge also requires an edge weight. The edge weight is a function of the position and velocity discontinuities between the parent node and the neighboring node, defined in this paper as:

$$w_e = w_d\|\mathbf{d}_1 - \mathbf{d}_2\| + w_v\|\mathbf{v}_1 - \mathbf{v}_2\| \tag{7}$$

where $n_1 = [d_1, v_1]$ is the final state along the arc associated with the parent node, $n_2 = [d_2, v_2]$ is the initial state on the arc described by the neighboring node, and w_d and w_v are the weights for the position and velocity components, respectively. A smaller edge weight represents smaller state discontinuities between the final state along the arc associated with the parent node and the initial state along the arc associated with the neighboring node. This edge weight serves as the cost function the search algorithm minimizes.

After an edge is added to each node, the roadmap's connectivity is assessed. The roadmap is considered to be connected if every node belongs to the same component of the graph, i.e. there exists at least one possible path from the starting node(s) to the final node(s). If the roadmap contains more than one component, edges are incrementally added one at a time to each node until the roadmap is connected. The connectivity of a graph is determined by evaluating the eigenvalues of the graph Laplacian matrix, an $n \times n$ matrix based on the number of neighbors each node has. If a graph is connected, the zero eigenvalue is isolated.[14] If no remaining valid edges exist, meaning the position difference is too great between the remaining nodes that are not already neighbors, then one randomly sampled node is added per trajectory. Then for all the nodes in the updated roadmap, each parent node is connected to its closest neighbor that it is not already connected to. This process is repeated until the roadmap is connected.

Once the roadmap is covered and connected, the desired initial and final periodic orbits are discretized into 100 states evenly spaced in time and added to the roadmap as additional nodes. For each node previously on the roadmap, possible edges to the initial and final orbit are constructed. An edge is added from the state on the periodic orbit to the node if the position difference is within the maximum arc length tolerance. Each edge also has an edge weight, defined by the same weighting function used to generate the roadmap. After the roadmap is covered and connected, it is considered to be complete and is used to construct an initial guess trajectory.

Phase 3: Query Phase

Once the roadmap is completed, it is searched to produce an initial guess for a transfer. A best first search algorithm is implemented to produce an efficient search of the roadmap. The roadmap may be searched multiple times for a variety of starting and ending configurations to produce multiple initial guesses for transfers. Each initial guess has the lowest total cost for the specific start and goal nodes selected, and is then corrected to produce a natural, heteroclinic connection. The selected discretized states along the initial and final orbit are identified as the start node(s) and goal node(s) within the roadmap. The start node with the lowest weight for one of its edges is the first node explored. Since the roadmap is searched using only the cost from the start to the current node, $g(n)$, and is not estimating the cost from the current node to the goal, such that $h(n) = 0$, Dijkstra's algorithm is implemented. Dijkstra's algorithm searches the completed roadmap for a sequence of nodes with the lowest total edge cost, $f(n) = g(n)$. This approach produces an initial guess

with the smallest cumulative discontinuities across the entire transfer. Once the initial guess has been constructed, a multiple shooting corrections scheme is applied to construct natural transfers between an L_1 and L_2 Lyapunov orbit.

RESULTS

PRM is used to construct initial guesses for transfers between an L_1 periodic orbit and an L_2 periodic orbit in the Earth-Moon CR3BP. First, natural transfers at a single Jacobi constant are explored. A planar transfer is constructed from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit at $C_J = 3.15$, where a direct heteroclinic connection exists.[15] A second roadmap is constructed at this energy level, using additional dynamical structures to reflect a larger portion of the solution space. This extended roadmap is searched for natural transfers in both directions, producing two transfers from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit and two transfers from an L_2 Lyapunov orbit to an L_1 Lyapunov orbit. The second scenario focuses on constructing multiple initial guesses from one generated roadmap reflecting states along periodic orbits and hyperbolic invariant manifolds with a range of $C_J = 3.13 - 3.15$. The roadmap is searched for transfers from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit at a single Jacobi level. The constructed transfers are then corrected via a multiple shooting corrections method to recover a continuous transfer in the CR3BP.

Parameters Governing Roadmap Construction

There are three parameters required to construct a roadmap in the CR3BP in this analysis: the maximum arc length, the position weight, and the velocity weight. The maximum arc length is a predefined parameter used to construct the roadmap that is selected based on the environment. The arc length dictates three factors in the roadmap: 1) it is used to find the propagation time for each node, such that all nodes have the same fixed arc length; 2) it dictates the maximum distance between two nodes to form an edge; and 3) it determines the neighborhood area covered by each node used for the coverage analysis. If the Euclidean distance between the final state on an arc associated with a parent node to the initial state on an arc associated with a neighboring node is less than the maximum arc length, the two nodes may be connected to form an edge. To determine an appropriate arc length for a transfer between an L_1 Lyapunov orbit and an L_2 Lyapunov orbit, the maximum arc length is computed as a percentage of the distance from L_1 to L_2. In this analysis, the maximum arc length for edges and the fixed arc length for nodes is 5% of the distance from L_1 to L_2. The only other predefined parameters selected for this analysis are the weights used in the edge weight function, displayed in Eq. 6. The position and velocity weights used for the edge weight function are selected such that the position and velocity components generally posses a similar order of magnitude. Since all transfers analyzed for this paper are in the Earth-Moon system, all parameters are constant across all scenarios. A summary of the selected values for these parameters governing roadmap construction is displayed in Table 1.

Table 1. Selected parameters governing roadmap generation

Parameter	Value
Maximum arc length	0.016 [n.d.]
Position weight, w_d	1
Velocity weight, w_v	0.3

Transfers from an L_1 Lyapunov to L_2 Lyapunov Orbit at $C_J = 3.15$

To construct an initial guess transfer from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit in the Earth-Moon CR3BP, the solution space is first computed. Each periodic orbit is computed via a single shooting corrections method, constraining the Jacobi constant to 3.15.[16] The periodic orbits, discretized into 100 states that are evenly spaced in time, serve as the set of initial and final nodes. Then, 50 trajectories are generated along the unstable manifold associated with the L_1 Lyapunov orbit and 50 trajectories along the stable manifold associated with the L_2 Lyapunov orbit. To sample the nodes of the roadmap, one node is sampled along an individual precomputed trajectory. The arc length of each trajectory is calculated via integration. A value is then sampled uniformly within the total arc length of the trajectory. The corresponding state along the trajectory serves as the initial state for the node. Then the node, beginning at that initial state, is integrated for the predefined arc length. Typically one of the hardest parameters to select to govern constructing a roadmap is the number of nodes to create an efficient representation of the solution space, without over sampling. The number of nodes for 100 successful simulations, those that were able to construct an initial guess transfer and corrected to a natural continuous transfer, is displayed in Figure 5. The mean number of each plot is denoted by the red line, the left and right edges of the box indicate the 25th and 75th percentiles, respectively, and additional data points are encompassed by the dashed black lines. The average number of nodes for these simulations was 21.92 nodes per trajectory. The average number of neighbors, forming the edges from each node, was 6.36 per node. This demonstrates the appropriate number of nodes and edges required to construct a sufficient roadmap depends on the quality of sampling performed.

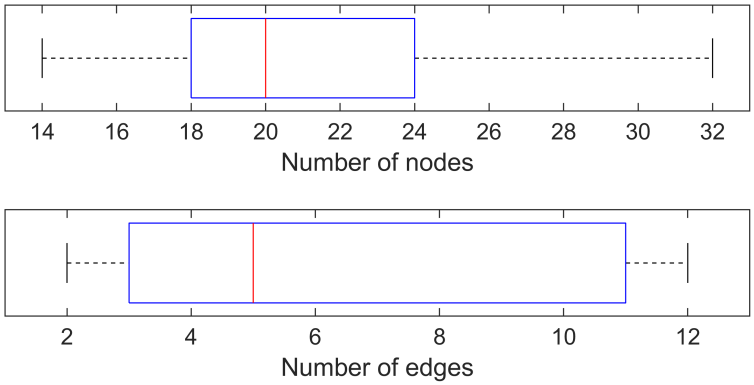

Figure 5. The average number of nodes (top) and the average number of edges per node (bottom) over 100 successful roadmaps.

The computational time required to construct the roadmap is significantly higher than the time required to search the roadmap.[8] Once the roadmap is generated, a search is performed multiple times for a variety of search queries, such as from periodic orbits at various energy levels. For this example, only one search was performed on the constructed roadmap from the L_1 periodic orbit to the L_2 periodic orbit, both at $C_J = 3.15$. The time required to generate a roadmap and search the corresponding graph for an initial guess is displayed in Figure 6. The mean number of each plot is denoted by the red line, the left and right edges of the box indicate the 25th and 75th percentiles, respectively, and additional data points are encompassed by the dashed black lines. The

outliers for this data are plotted in red plus symbols. Across 100 simulations, the average time to construct the roadmap was 100.71 seconds and the average time to search the corresponding graph was 1.99 seconds. The time required to construct the roadmap is more computationally expensive compared to typical path planning problems because the nodes and edges are both added incrementally, as such the coverage and connectivity of the graph are checked more frequently. By letting the algorithm intuitively decide how many nodes and edges to add to the roadmap, this reduces the burden on the user to have a priori knowledge of the environment that would typically be required to make an effective choice for these predefined parameters. This trade off is worth the extra required computational time.

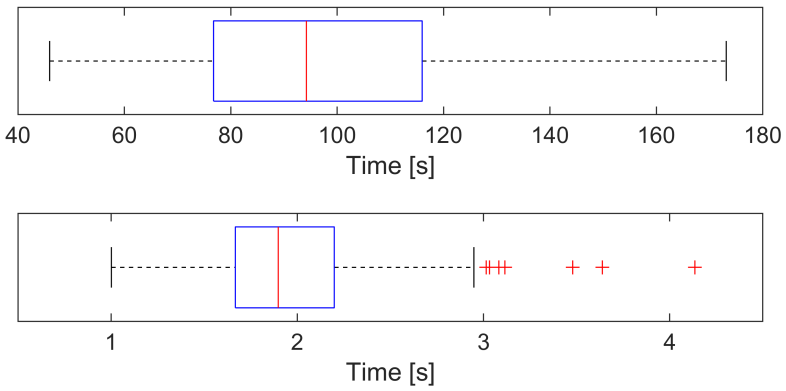

Figure 6. The average computational time in seconds to construct the roadmap (top) and to search for an initial guess (bottom) over 100 successful roadmaps.

To demonstrate designing an initial guess transfer via PRM, one example simulation is explored in detail. First, the nodes and edges for the constructed roadmap are displayed separately in Figure 7 and Figure 8, respectively. In Figure 7, the initial state for each node is indicated in blue, the resulting trajectory segment propagated until the fixed arc length is plotted in light blue, and the final state on each arc associated with a node is displayed in pink. In Figure 8, the edges are displayed as black dashed lines from the initial state defined by a parent node to the initial state for the neighboring node. However, neighbors are selected within the algorithm using the discontinuity between the final state on a parent node and the initial state on the neighboring node. The initial state for each node is only displayed in blue for simplicity. The complete roadmap, containing both nodes and edges represented in these two figures using the color scheme described above, is displayed in Figure 9. Once the roadmap is generated, Dijkstra's algorithm is used to construct an initial guess for a transfer. The discontinuous initial guess, corresponding to the roadmap displayed in Figure 9, is displayed in Figure 10a). For each node in the initial guess, the initial state is indicated as a blue circle and the associating trajectory segment plotted in blue. To recover a continuous natural transfer at $C_J = 3.15$, the initial guess is corrected via multiple shooting. The natural transfer is displayed in Figure 10b). The geometry of the discontinuous initial guess is slightly different than the natural transfer at the beginning and end of the transfer where the trajectory departed the L_1 Lyapunov orbit and arrived to the L_2 Lyapunov orbit, but closely resembles the natural transfer for the rest of the initial guess. As more nodes are sampled along additional fundamental solutions, the geometries would become more similar. However, the constructed initial guess lies sufficiently close to a continuous, natural transfer in the Earth-Moon CR3BP for corrections.

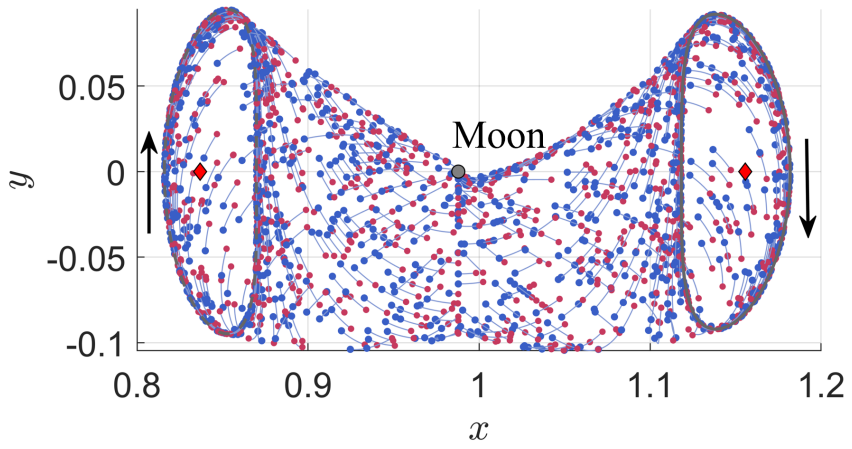

Figure 7. Randomly sampled nodes for a completed roadmap at $C_J = 3.15$.

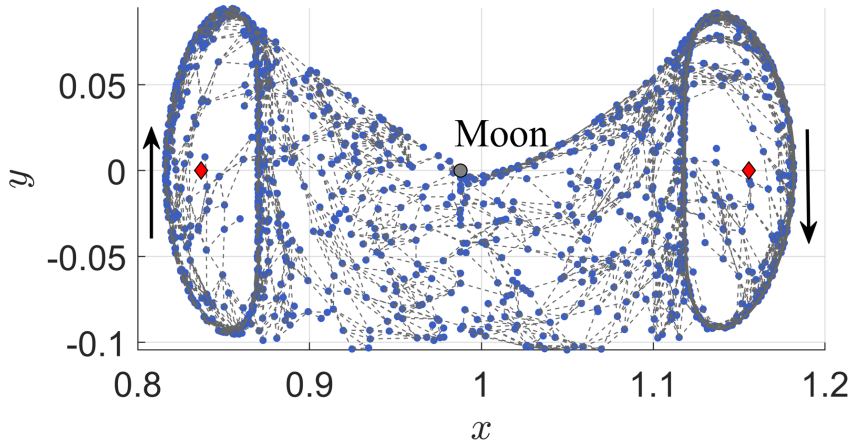

Figure 8. Constructed valid edges for a completed roadmap at $C_J = 3.15$.

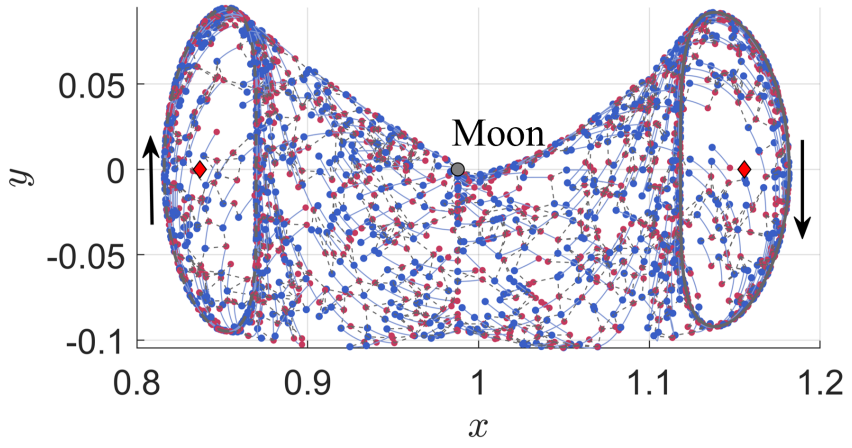

Figure 9. An example completed roadmap for $C_J = 3.15$.

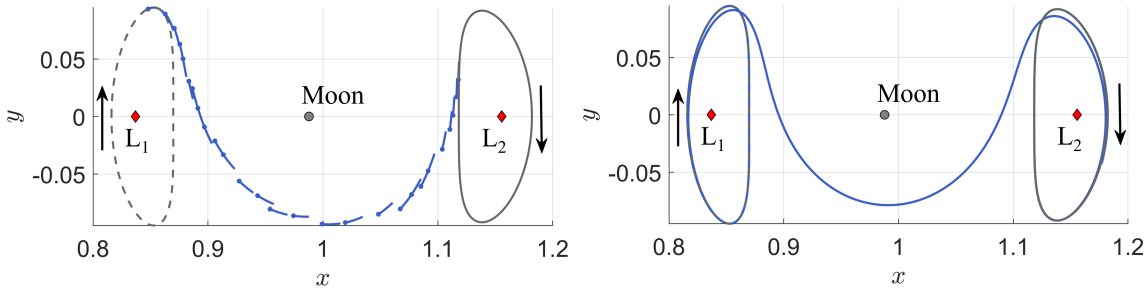

Figure 10. a) The initial guess constructed from the roadmap in Figure 9 and b) the corrected continuous transfer at $C_J = 3.15$

To explore all natural transfers that exist at this energy level, the roadmap is expanded to summarize additional fundamental solutions and cover a larger portion of the solution space at $C_J = 3.15$. The roadmap is constructed using L_1 and L_2 Lyapunov orbits and 50 trajectories along each of the associated stable and unstable invariant manifolds. Nodes are sampled from these 200 trajectories and edges are formed, both using the same construction strategy as in the previous example. An example of this extended roadmap is displayed in Figure 11a). To construct multiple initial guesses that vary geometrically, a single state along the L_1 and L_2 Lyapunov orbit are selected as the start and goal node within the roadmap rather than allowing the initial guess to start and end anywhere along the orbits. The roadmap is also then explored with a state along the L_2 Lyapunov orbit selected as the start node and a state along the L_1 Lyapunov orbit selected as the goal node. By searching the roadmap from a variety of start and goal nodes, all natural solutions with this geometry that exist at $C_J = 3.15$ to and from each Lyapunov orbit are recovered. The recovered transfers are displayed in Figure 11b): two transfers from L_1 to L_2 and two transfers from L_2 to L_1, each with a distinct geometry. These transfers are confirmed to lie close to heteroclinic connections, identified via Poincaré mapping.

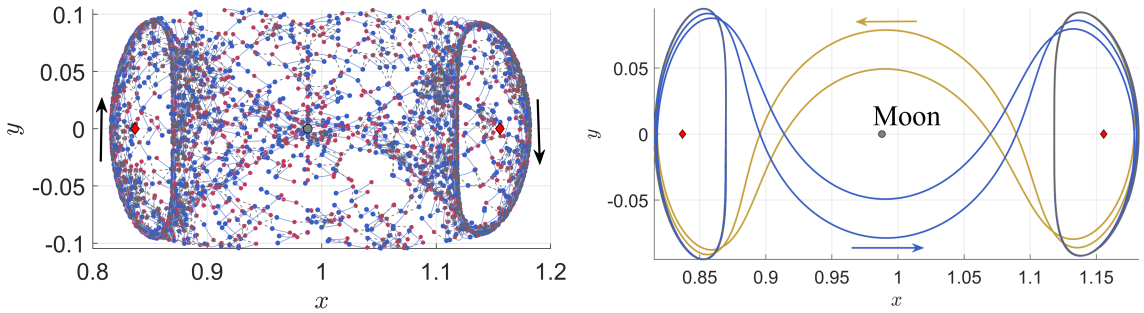

Figure 11. a) An extended completed roadmap for $C_J = 3.15$ and b) all corrected natural transfers at $C_J = 3.15$

Multiple Energy Level Planar Transfers

A key advantage of path planning techniques is the capability to search a roadmap for multiple queries rapidly. During trajectory design, constraints and mission goals change frequently. In traditional trajectory design methods, this means redesigning a trajectory often. With PRM, once the roadmap has been generated, changing the start and goal configurations and constructing a new

path is simple and efficient. In this section, a roadmap is generated from trajectories at multiple energy levels rather than sampling nodes at one desired energy level. Then, initial guess transfers are constructed using the same roadmap, each with different start and goal configurations: L_1 and L_2 Lyapunov orbits at different energy levels. This roadmap is constructed in the Earth-Moon CR3BP, near the vicinity of the Moon. As such, the predefined roadmap parameters in Table 1 are used.

The roadmap for this example is constructed from Lyapunov orbits and trajectories along the associated hyperbolic invariant manifolds from a range of Jacobi constants in the range of $C_J = [3.13, 3.15]$. The sampling subset for the roadmap, an L_1 Lyapunov orbit and its associated unstable manifold, and a L_2 Lyapunov orbit and its associated stable manifold, are computed at three distinct energy levels: $C_J = 3.13, C_J = 3.14$, and $C_J = 3.15$. 50 trajectories along each of the 3 unstable manifolds and 3 stable manifolds are computed, totaling 300 trajectories in the sampling subset. From these trajectories, the same sampling scheme as described above is employed. The main difference being that now nodes are sampled from a variety of energy levels. Nodes are sampled until the area covered by the initial state from each node covers the same area as the input trajectories in the phase space. Once the roadmap is covered, edges between neighboring nodes are added until the roadmap is connected. An example roadmap generated from this subset of trajectories is displayed in Figure 12.

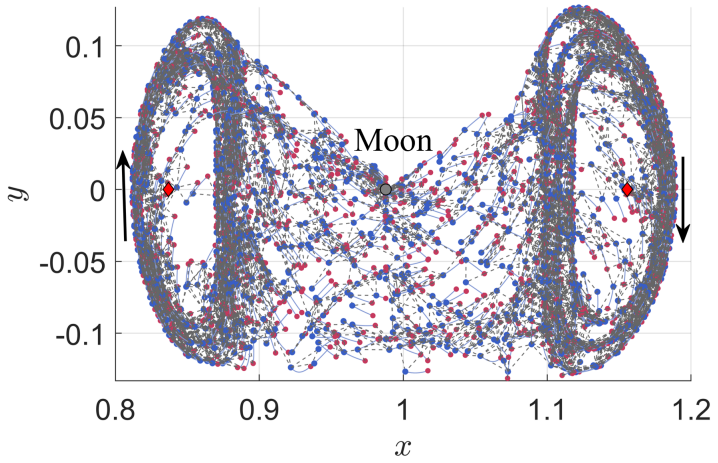

Figure 12. An example roadmap generated for $C_J = [3.13 - 3.15]$.

To construct multiple initial guesses for transfers from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit in the Earth-Moon CR3BP using the same roadmap, three search queries are used with varying start and goal configurations. For each distinct energy level used to generate the sampling subset, one search is performed. The start nodes and goal nodes are the discretized states along the L_1 and L_2 Lyapunov orbits at the desired energy level. Then, each initial guess is corrected via multiple shooting and constrained to the energy level of the start and goal orbits energy level. The initial guess transfers and their corresponding corrected transfers are displayed in Figure 13. Similar to the previous example, the geometries of each initial guess vary at the departure from the L_1 orbit and arrival to the L_2 orbit, but each initial guess found different transfers for the desired energy level. The three search queries for each initial guess are constructed from the same roadmap, but each distinct initial guess is successfully corrected to produce a continuous transfer at the desired energy level. The Jacobi constant for each node selected in each constructed initial guess is displayed in

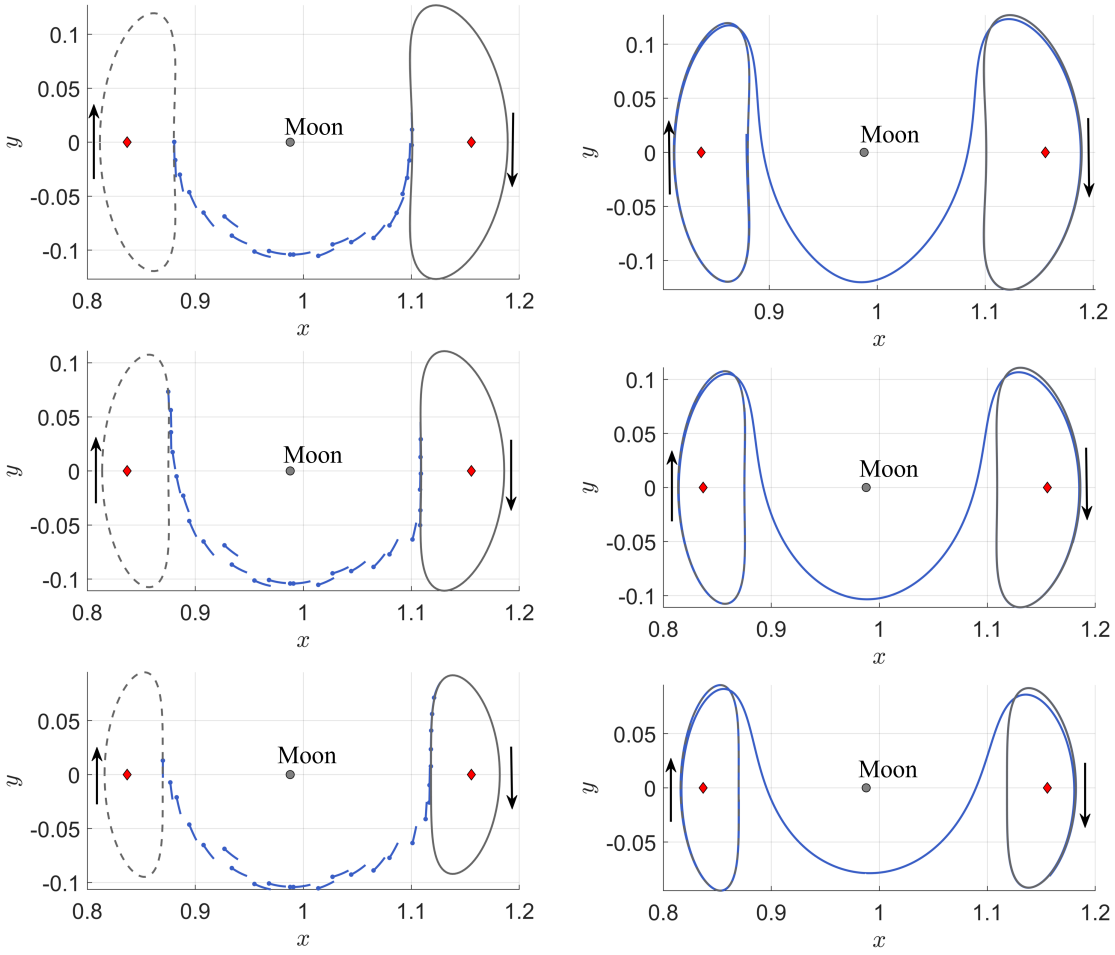

Figure 13. Three initial guess transfers and the corresponding corrected transfer at: a) $C_J = 3.13$**, b)** $C_J = 3.14$**, and c)** $C_J = 3.15$**.**

Figure 14. Because the start and goal orbit were different for each search query, the search algorithm selected 3 different first nodes in each guess as well as 3 different ending nodes. Although the first and last nodes were all different, all 3 initial guesses selected the same nodes at the Moon hyperplane. This shows the nodes selected at the hyperplane were the lowest cost for all three transfers, regardless of the desired energy level, in the roadmap. This example demonstrates the roadmap can be extended to summarize additional fundamental solutions and recover a larger subset of natural transfers for a given environment.

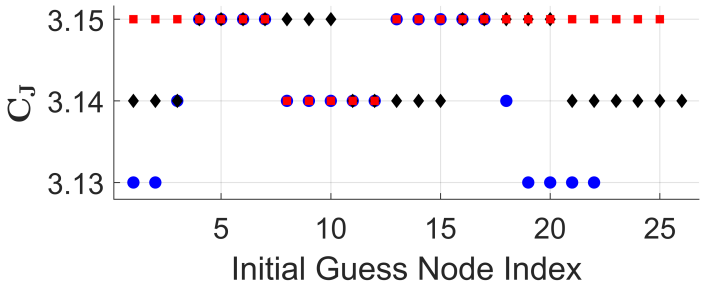

Figure 14. C_J **levels across initial guess transfers depicted in Figure 13 at** $C_J = 3.13$ **(blue),** $C_J = 3.14$ **(black), and** $C_J = 3.15$ **(red).**

CONCLUSION

Trajectory design is a time-consuming process that often relies heavily on a human-in-the-loop, with current state of the art approaches limiting the rapid exploration of the solution space. However, path planning techniques may offer a potential approach to efficiently summarize and explore complex environments. In this paper, probabilistic roadmap generation is employed in the Earth-Moon CR3BP to summarize a part of the solution space near the vicinity of the Moon and construct initial guesses for transfers between an L_1 Lyapunov orbit and an L_2 Lyapunov orbit; the initial guesses are then corrected to produce a continuous natural transfer.

Using PRM in the CR3BP requires three phases: the preprocessing phase, the learning phase, and the query phase. During preprocessing phase, fundamental solutions such as periodic orbits and their associated manifolds are computed to represent the desired solution space. States along these solutions are used as the data subset from which nodes are sampled from. During the learning phase, a weighted, directed graph is generated. Nodes are sampled from the data subset incrementally until the desired solution space is sufficiently covered and edges are also added incrementally until the roadmap is connected. An edge weight representing the discontinuity between nodes is assigned to each edge to serve as the cost function the search algorithm minimizes. Finally, the roadmap is searched using Dijkstra's algorithm for an initial guess from a user defined start node(s) and goal node(s) during the query phase.

First, a planar transfer at $C_J = 3.15$ is explored to generate a roadmap. This roadmap is searched once to produce a discontinuous initial guess and then corrected to recover a continuous transfer at this energy level. Then, the roadmap is extended to summarize a wider variety of fundamental solutions and explore a larger portion of the solution space. All natural transfers at $C_J = 3.15$, i.e., two from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit and two from an L_2 Lyapunov orbit to an L_1 Lyapunov orbit, are recovered using this extended roadmap by selecting an initial start node and initial goal node along the desired periodic orbits. A second scenario to construct initial guesses

for transfers from a range of energy levels, $C_J = 3.13 - 3.15$, is explored. One roadmap is constructed using fundamental solutions at three distinct energy levels. Then, the roadmap is searched using three separate queries with different start and goal configurations. Three initial guesses are constructed and corrected to recover three natural transfers from an L_1 Lyapunov orbit to an L_2 Lyapunov orbit at three energy levels. Although the geometry differed slightly in both scenarios between the initial guesses constructed and the natural transfers recover, each roadmap successfully summarized and explored the desired solution space to construct the desired continuous transfers. This successful preliminary example demonstrated in this paper motivates the additional exploration of applying the roadmap generation process to a larger variety of trajectory design scenarios in multi-body systems.

ACKNOWLEDGMENT

This research is being completed at the University of Colorado Boulder. It is supported by a NASA Space Technology Graduate Research Opportunity.

REFERENCES

[1] P. Svestka and M. H. Overmars, *Robot Motion Planning and Control*. Berlin: Springer, Berlin, Heidelberg, 1st ed., 1998.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 1st ed., 2006.

[3] K. Lynch, G. Kantor, H. Choset, L. Kavraki, S. A. Hutchinson, W. Burgard, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 1st ed., 2005.

[4] A. Das-Stuart, K. Howell, and D. Folta, "A Rapid Trajectory Design Strategy for Complex Environments Leveraging Attainable Regions and Low-Thrust Capabilities," Adelaide, Australia, 68th International Astronautical Congress, September 2017.

[5] J. A. Starek, E. Schmerling, G. D. Maher, B. W. Barbee, and M. Pavone, "Real-Time, Propellant-Optimized Spacecraft Motion Planning under Clohessy-Whilshire-Hill Dynamics," Big Sky, MT, IEEE Aerospace Conference, March 2016.

[6] E. Trumbauer and B. Villac, "Heuristic Search-Based Framework for Onboard Trajectory Redesign," *Journal of Guidance, Control and Dynamics*, Vol. 37, No. 1, January-February 2014, 10.2514/1.61236.

[7] G. A. Tsirogiannis, "A graph based methodology for mission design," *Celestial Mechanics and Dynamical Astronomy*, Vol. 114, 2012, 10.1007/s10569-012-9444-9.

[8] N. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An Obstacle Based PRM for 3D workspaces," *Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics*, Wellesley, MA, 1998.

[9] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, *Dynamical Systems, The Three-Body Problem and Space Mission Design*. Marsden Books, 3rd ed., 2006.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 1999.

[11] L. E. Kavraki and S. M. LaValle, *Motion Planning*. Springer Handbook of Robotics. Springer, Berlin, Heidelberg, 2008, 10.1007/978-3-540-30301-5_6.

[12] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice-Hall, 3rd ed., 2009.

[13] J.-H. Park and T.-W. Yoon, "Maximizing the Coverage of Roadmap Graph for Optimal Motion Planning," *Hindawi*, Vol. 2018, November 2018, 10.1155/2018/9104720.

[14] R. Olfati-Saber, A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, Vol. 95, No. 1, January 2007, 10.1109/JPROC.2006.887293.

[15] E. Canalias and J. J. Masdemont, "Homoclinic and Heteroclinic Transfer Trajectories between Lyapunov Orbits in the Sun-Earth and Earth-Moon Systems," 10 2007.

[16] N. Bosanac, *Leveraging Natural Dynamical Structures to Explore Multi-Body Systems*. PhD thesis, Purdue University, IN, 2016.