(Preprint) AAS 25-783

LONG-TERM SPACECRAFT TRAJECTORY PREDICTION USING BEHAVIORAL MOTION PRIMITIVES

Austin Bodin[*], Natasha Bosanac[†], and Cole Gillespie[‡]

This paper leverages behavioral motion primitives for spacecraft trajectory prediction in cislunar space. A behavioral motion primitive summarizes trajectories with a similar geometry and is labeled by the associated behaviors (e.g., maneuvering objective and intent) and spacecraft parameters. The volume spanned by these geometrically similar trajectories is labeled as the region of existence of each primitive and approximated using voxels. Uncertain state estimates are projected onto these regions of existence for short-term predictions. Sequences of composable primitives, with overlapping regions of existence in the configuration space, are then used to generate digestible long-term trajectory predictions.

INTRODUCTION

Cislunar space is of growing interest for supporting commercial infrastructure, scientific exploration, and defense applications. The associated spacecraft could possess a wide array of behaviors composed of intents, i.e., itineraries; maneuvering objectives, i.e., control profiles; and spacecraft properties, e.g., propulsion system and mass. The range of possible intents, maneuvering objectives, and spacecraft properties produces a diverse array of motions that a spacecraft could follow within the chaotic dynamical environment of cislunar space. Furthermore, similar trajectory geometries can be generated through distinct combinations of spacecraft properties and behaviors. As a result, with limited or no information about the behaviors and capabilities of an observed spacecraft, trajectory prediction from uncertain state estimates becomes a challenging task that can produce an overwhelming amount of information for an operator or analyst.

Initial orbit determination (IOD) is a critical task for space operations. Foundational methods such as Laplace's method, Gauss' method, Double r-iteration, and Gooding's method[1,2] construct orbit fits that often rely on concepts or mathematical relationships from the two-body problem, which can lead to poor approximations in a multi-body system. To refine the state estimate and provide state uncertainty, large numbers of these measurements are often used in a batch estimation method such as weighted nonlinear least squares.[3] The IOD estimate is then used to predict possible trajectories of the spacecraft. Existing techniques including Monte Carlo analysis, which relies on repeated random sampling to numerically generate trajectories,[4] can impose a high computational burden when there is limited information on the behavior and properties of the spacecraft. Linear

[*]Graduate Research Assistant, Ann & H.J. Smead Department of Aerospace Engineering Sciences, Colorado Center for Astrodynamics Research, University of Colorado Boulder, Boulder, CO, 80303.
[†]Associate Professor, Ann & H.J. Smead Department of Aerospace Engineering Sciences, Colorado Center for Astrodynamics Research, University of Colorado Boulder, Boulder, CO, 80303.
[‡]Graduate Research Assistant, Ann & H.J. Smead Department of Aerospace Engineering Sciences, Colorado Center for Astrodynamics Research, University of Colorado Boulder, Boulder, CO, 80303.

mean and covariance propagation, as used in various Kalman filtering implementations, can struggle when distinct types of motion emerge over long time intervals.[5] To address this, Sorenson and Alspach have extended linear uncertainty propagation to some nonlinear systems using Gaussian sums to represent non-Gaussian probability distributions.[6] Although these heritage methods have proven valuable for IOD in dynamical environments where the gravity field of a single celestial body is dominant, they have limited applicability in cislunar space. Furthermore, these approaches may result in a large amount of generated information.

Complex path and movement prediction problems have been encountered across a diverse array of disciplines. In robotics, the concept of motion primitives has been used to represent fundamental building blocks of motion.[7] These motion primitives can be extracted from groups of similar motion within a set of observational or simulated data.[8–10] A library of motion primitives then supplies a discrete and digestible summary of the solution space. By sequencing composable motion primitives, analogous to assembling building blocks with compatible connections, more complex paths can be rapidly generated.[11] Due to their capability to summarize distinct types of paths or movements, motion primitives have been useful for supplying rapid and digestible path and behavioral predictions. For example, Habibi, Jaipuria, and How have used motion primitives to predict future pedestrian motions.[12] Edelhoff et al. also used motion primitives to extract behavioral models from animal movement observations and identify behavioral changes.[13]

Motion primitives have been used for spacecraft trajectory design in multi-body gravitational systems. Smith and Bosanac have used motion primitives to summarize a set of arcs with a similar geometry in the circular restricted three-body problem (CR3BP).[14] These groups of geometrically similar arcs have been extracted using clustering, with substantial improvements recently developed by Bosanac as well as Gillespie, Miceli, and Bosanac.[15,16] Distinct motion primitives separate motions with distinct geometries. Smith and Bosanac also defined a graph-based procedure to identify sequences of motion primitives that produce geometrically distinct initial guesses for spacecraft trajectory design.[14] Miceli, Bosanac, and Karimi have extended this work to improve the graph construction process, incorporate constraints, and improve the graph search algorithm to produce a more diverse array of primitive sequences.[17,18] Leveraging this foundation, Gillespie, Miceli, and Bosanac have introduced behavioral motion primitives to also encode the array of behaviors, maneuvering objectives, and spacecraft parameters that produce each group of geometrically similar arcs for natural and thrust-enabled motions in cislunar space.[16]

This paper focuses on using behavioral motion primitives for long-term spacecraft trajectory prediction in cislunar space. First, this paper leverages the diverse library of behavioral motion primitives generated by Gillespie, Miceli, and Bosanac.[16] Then, the region of the phase space encompassed by arcs that geometrically resemble each primitive, labeled its region of existence, is approximated using voxels. Initial state estimates are then projected onto the library of behavioral motion primitives: any primitives with regions of existence that overlap with an uncertainty ellipsoid centered on the state estimate supply short-term motion predictions. Next, recent work by Miceli and Bosanac[19] is extended to construct a graph of behavioral motion primitives. This graph captures the connections between primitives that are sequentially composable in the configuration space, estimated via intersections of the voxel approximation of their regions of existence. Searching this graph produces primitive sequences that are used to rapidly generate a digestible summary of long-term possible trajectory predictions. This new approach to trajectory prediction is demonstrated by exploring generating possible long-term future motions of a spacecraft from an uncertain state estimate in the planar Earth-Moon CR3BP, assuming only impulsive maneuvers.

BACKGROUND

Dynamical Model

The Circular Restricted Three-Body Problem (CR3BP) is used to model the motion of a spacecraft. This dynamical model assumes that the mass of the spacecraft is negligible compared to the two primary bodies, the Earth and the Moon, which follow circular orbits.[20] The primary bodies are also modeled with the same gravity fields as point masses with masses M_1 and M_2, respectively.[20]

State vectors are specified in nondimensional form using a rotating frame. The rotating frame is defined with an origin at the center of mass of the Earth-Moon system. The axes $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$ are defined with $\hat{\boldsymbol{x}}$ directed from the Earth to the Moon, $\hat{\boldsymbol{z}}$ aligned with the orbital angular momentum vector of the primaries, and $\hat{\boldsymbol{y}}$ completes the right-handed, orthogonal triad.[20] Quantities are also nondimensionalized using characteristic parameters for length (l^*), mass (m^*), and time (t^*): $l^* = 384,400$ km is the distance between the Earth and Moon, $m^* = 6.046 \times 10^{24}$ kg is the total system mass, and $t^* = 3.752 \times 10^5$ s produces a nondimensional period of the primaries equal to 2π.

The equations of motion for the spacecraft are typically expressed in the Earth-Moon rotating frame. The nondimensional state of the spacecraft is first defined as $\boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^{\mathrm{T}}$. Then, the equations of motion are written as[20]

$$\ddot{x} = 2\dot{y} + \frac{\partial U^*}{\partial x}, \quad \ddot{y} = -2\dot{x} + \frac{\partial U^*}{\partial y}, \quad \ddot{z} = \frac{\partial U^*}{\partial z}, \tag{1}$$

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 to the primaries, and $U^* = 0.5(x^2 + y^2) + (1-\mu)/r_1 + \mu/r_2$. In the Earth-Moon system, the mass ratio of the primaries is $\mu \approx 0.01215$. The Jacobi constant is an integral of motion equal to[20]

$$C_J = 2U^* - \dot{x}^2 - \dot{y}^2 - \dot{z}^2. \tag{2}$$

This energy-like quantity supplies insight into allowable regions of motion.[20] In this dynamical model, the five libration points, L_1-L_5 are stationary solutions in the system.[20] Periodic orbits are solutions that repeat after a finite time in the rotating frame. Finally, hyperbolic invariant manifolds asymptotically depart or approach a libration point, periodic orbit, or quasi-periodic orbit.

Differential Geometry

Differential geometry supports the analysis of nonlinear trajectories. At any instant of time, the position, velocity, and acceleration vectors are equal to $\boldsymbol{r}(t) = [x(t),\, y(t),\, z(t)]^{\mathrm{T}}$, $\boldsymbol{v}(t) = [\dot{x}(t),\, \dot{y}(t),\, \dot{z}(t)]^{\mathrm{T}}$, and $\boldsymbol{a}(t) = [\ddot{x}(t),\, \ddot{y}(t),\, \ddot{z}(t)]^{\mathrm{T}}$, respectively. Over the time interval $t \in [t_0,\, t_f]$, a trajectory propagated from a reference initial state traverses a distance equal to the arclength[21]

$$s_p \;=\; \int_{s_0}^{s_f} ds_p \;=\; \int_{t_0}^{t_f} \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}\, dt. \tag{3}$$

The velocity hodograph also traces out a curve with the following distance:

$$s_v \;=\; \int_{s_0}^{s_f} ds_p \;=\; \int_{t_0}^{t_f} \sqrt{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2}\, dt, \tag{4}$$

equivalent to the cumulative change in velocity over the trajectory.

At any location along the trajectory, the curvature captures the extent to which the path curves away from a straight line. Curvature is calculated as[22]

$$\kappa(\bar{x}) = \frac{\sqrt{(\dot{x}\,\ddot{y} - \dot{y}\,\ddot{x})^2 + (\dot{z}\,\ddot{x} - \dot{x}\,\ddot{z})^2 + (\dot{y}\,\ddot{z} - \ddot{y}\,\dot{z})^2}}{(\dot{x}^2 + \dot{y}^2 + \dot{z}^2)^{3/2}}. \tag{5}$$

At maxima in curvature, the trajectory most rapidly changes direction. In the CR3BP, these maxima often occur close to apses relative to the meaningful reference points.[23]

Density-Based Clustering

Clustering partitions a dataset into groups of similar data points while separating dissimilar data. In this paper, density-based techniques are used to identify regions in a finite-dimensional feature space where data points exhibit sufficient local concentration. Specifically, two algorithms are employed: Density-Based Spatial Clustering of Applications with Noise (DBSCAN)[24] and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN).[25]

DBSCAN was introduced by Ester, Kriegel, Sander, and Xu to detect arbitrarily shaped clusters by linking points with overlapping neighborhoods.[24] A core point possesses at least m_{pts} points (including itself) that fall within its ϵ-neighborhood. Those points that lie within the ϵ-neighborhood of a core point but do not themselves satisfy the core point criterion are labeled border points. The remaining points are labeled as noise. Clusters in DBSCAN consist of all core points that are density-connected, reachable through a chain of overlapping core neighborhoods, and their associated border points. Because DBSCAN requires specifying both m_{pts} and ϵ, it is useful when these parameters can be selected heuristically. In this paper, MATLAB's *dbscan* function is employed.

HDBSCAN, developed by Campello, Moulavi, and Sander, extends DBSCAN by building a cluster hierarchy over varying density thresholds.[25] HDBSCAN begins by defining the mutual reachability distance between two points as the maximum of their pairwise distance and each point's core distance (the distance to its m_{pts}-th nearest neighbor). A minimum spanning tree constructed on this metric captures the data's intrinsic density structure. This tree is then condensed into a cluster hierarchy where each node represents a candidate cluster containing at least $m_{min,clust}$ points. When moving up the hierarchy, which corresponds to increasing the density level, clusters may shrink, split, or vanish. A stability criterion based on excess of mass is used to extract the most persistent clusters from this hierarchy, whereas points that fail to meet density requirements remain classified as noise.[25] Because HDBSCAN does not depend on a fixed neighborhood radius and can accommodate clusters of differing densities and sizes, it is particularly useful when ϵ is not known in advance. HDBSCAN is governed by two parameters: m_{pts} and $m_{min,clust}$.[25] An explicit reachability threshold ϵ may also be applied to constrain cluster formation.[26] In this paper, HDBSCAN is implemented via the Python *hdbscan* library.[27]

Numerically Correcting Trajectories

Multiple shooting is used to recover continuous trajectories from discontinuous initial guesses. In this paper, a free variable vector $\boldsymbol{V}$ is defined as

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{x}_{1,0} & \Delta T_1 & \boldsymbol{x}_{2,0} & \Delta T_2 & \dots & \boldsymbol{x}_{n,0} & \Delta T_n \end{bmatrix}^{\mathrm{T}} \tag{6}$$

where $\boldsymbol{x}_{i,0}$ and ΔT_i are the initial state and time of flight along the i-th arc, respectively. A constraint vector $\boldsymbol{F}_0(\boldsymbol{V})$ that only enforces continuity in the configuration space while allowing impulsive

maneuvers between subsequent arcs is defined as

$$\boldsymbol{F}_0(\boldsymbol{V}) = \left[\boldsymbol{r}_{1,f}^{\mathrm{T}} - \boldsymbol{r}_{2,0}^{\mathrm{T}}, \boldsymbol{r}_{2,f}^{\mathrm{T}} - \boldsymbol{r}_{3,0}^{\mathrm{T}}, ..., \boldsymbol{r}_{n-1,f}^{\mathrm{T}} - \boldsymbol{r}_{n,0}^{\mathrm{T}}\right]^{\mathrm{T}} \tag{7}$$

where $\boldsymbol{r}_{i,0}$ is the initial position vector along the ith arc and $\boldsymbol{r}_{i,f}$ is the final position along the ith arc. Additional boundary and path constraints can be added as needed, whereas full state continuity can be used if a maneuver is not allowed. Inequality constraints can be included by converting them to equality constraints through the addition of slack variables β_i in the free variable vector. Once any additional constraints are appended to $\boldsymbol{F}_0$ to form the full constraint vector $\boldsymbol{F}$, an initial guess for the free variable vector is iteratively updated via Newton's method. These updates continue until $|\boldsymbol{F}(\boldsymbol{V})| < \tau$, where τ is a specified tolerance. At each iteration, the derivatives of the constraints with respect to the free variables are calculated analytically.

Voxel Representations

A voxel is a three-dimensional analog of a pixel often used for computer graphics and medical or scientific visual analysis.[28] Voxels are typically cubes with uniform size used to discretize more complex shapes. This volume representation supports efficient rendering, simple intersection or overlap calculations, and variable resolution. Their positions can be defined through a spatial data structure (e.g. an octree or k-d tree), or each voxel can carry its own absolute coordinates.[28] Voxels can also be encoded with additional information, such as color, texture, or group identifiers.[28] In this paper, voxels are utilized to represent volumes of the phase space with uniform side lengths s_i and specified center positions $\boldsymbol{c}_i$, as depicted in Figure 1.

Graph Search Algorithms

Two fundamental search algorithms are Dijkstra's[29] and A*,[30] both of which are used in this work. These algorithms are used within weighted graphs with nonnegative edge weights to identify the shortest path from one node, the start node, to another node in the graph, the end node. Although Dijkstra's algorithm can identify the shortest path from a start node to all other reachable nodes, it is often terminated early to provide just the shortest path to a single end node.[29] In graph theory, the term "shortest path" is the path with the minimum cost when summing the weights of the traversed edges.[29] In general, the edge weights or costs in a graph can be any value that describes the ease of moving from one node to another.

The A* search algorithm identifies the shortest path between two nodes faster than Dijkstra's algorithm through the use of a heuristic.[30] The heuristic guides the search algorithm at each node through providing approximate knowledge of the cost to the end node. For this algorithm to work

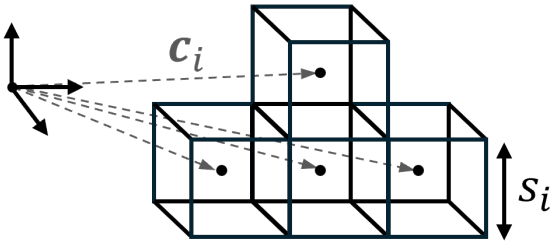


Figure 1: A group of voxels, each with a side length s_i and absolute center positions $\boldsymbol{c}_i$.

properly, the heuristic must be "admissible," i.e., $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost from node n to the end node.[30] As $h(n)$ approaches 0, A* explores more of the graph and effectively behaves like Dijkstra's algorithm. As $h(n)$ approaches $h^*(n)$, A* explores fewer nodes, traversing only those necessary to reach the goal.

Yen's algorithm identifies the K-shortest loop-less paths between a start node and an end node in a weighted graph with non-negative edge costs.[31] This graph search algorithm begins by computing the first shortest path using Dijkstra's algorithm. Additional candidate paths $k = 2, 3, \ldots, K$ are generated using each node i in the $(k-1)$-th path: 1) the segment from the start node to node i is defined as a root path, 2) edges that would recreate any of the previous $(k-1)$ root segments up to i are temporarily removed, 3) a spur path is computed from i to the goal, and 4) that spur path is concatenated to the root to form a complete candidate path. These candidates are stored and the lowest-cost candidate is selected as the next shortest path. This spur-and-root procedure repeats until the top K loop-less paths have been identified.

TECHNICAL APPROACH

Building a Motion Primitive Library

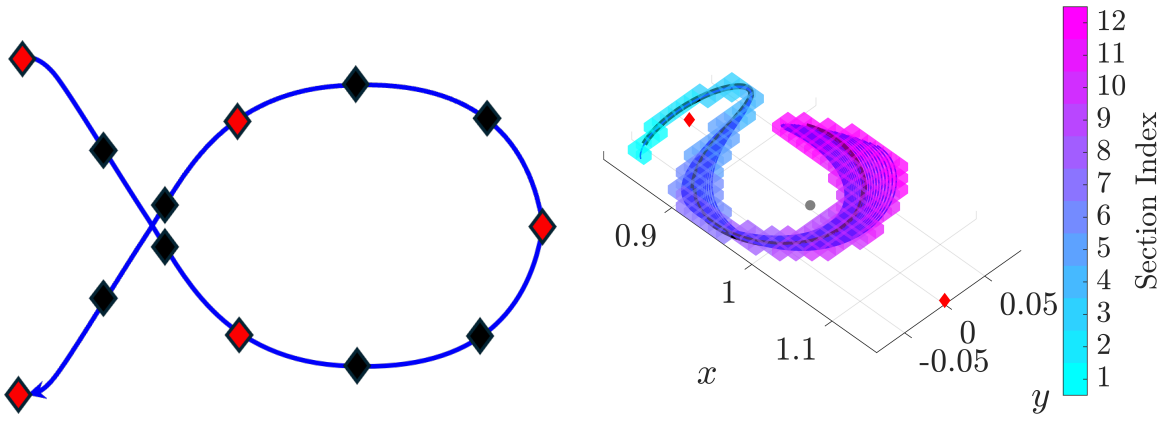
A library of motion primitives is generated to discretely summarize the continuous solution space. This step closely follows the procedure previously developed by Gillespie, Miceli, and Bosanac[16] and a brief overview is presented here. In this proof of concept, the motion primitives are generated to summarize natural arcs along the stable and unstable manifolds of Lyapunov orbits. Although behavioral labels are not employed in this preprint, their use is an avenue of ongoing work.

The stable and unstable manifolds of Lyapunov orbits are numerically approximated. First, each orbit is discretized into 500 states that are equally distributed in the arclength. Then, these states are perturbed to produce initial conditions that lie in the stable or unstable eigenspaces. Each initial condition is propagated for a duration equal to $\Delta t_{prop} = \Delta t_{pertdoub} +$ 3 months. In this expression, $\Delta t_{pertdoub}$ is the perturbation doubling time, calculated from the magnitude of the unstable eigenvalue of the monodromy matrix.[32] However, propagation terminates early upon impact with either the Earth or Moon. During propagation, the curvature maxima along the trajectories are recorded.

Arcs are then sampled as "building-block" sized segments of each set of approach or departure trajectories.[16] For a trajectory that possess at least three maxima in the curvature, the initial state, each maximum in curvature, and any impact states supply initial conditions for the arc. Each arc is then defined to span an additional three curvature maxima as well either a fourth maximum in curvature or the termination condition. If a trajectory possesses fewer than three maxima in the curvature, the window is shortened but the initial and final state definitions remain unchanged.

Each arc is then sampled geometrically.[16] First, the arc is sampled at its initial condition, any maxima in the curvature, and any termination conditions. In between these initial samples, two additional samples are added spaced evenly in arclength. Figure 2(a) depicts a conceptual trajectory sampled in this manner where the red diamonds mark the initial samples and the black diamonds locate the intermediate samples. The region in between each sample is designated a "section." For the example in Figure 2(a), the arc is divided into twelve sections. The red diamonds are curvature maxima and the black diamonds are intermediate arclength samples. This sampling methodology ensures that samples are placed in geometrically meaningful regions.

These sampled states are used to create two feature vectors that supply a finite-dimensional de-



(a) (b)

Figure 2: (a) Conceptual trajectory discretized into sections to encompass five maxima in the curvature. (b) The region of existence for a motion primitive departing an L_1 Lyapunov orbit.

scription of a continuous arc.[16,33] A shape-based feature vector $\boldsymbol{f}_s$ and position-based feature vector $\boldsymbol{f}_p$ are defined as

$$\boldsymbol{f}_s = [\hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2, ..., \hat{\boldsymbol{v}}_n]^{\mathrm{T}} \qquad \boldsymbol{f}_p = [\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_n]^{\mathrm{T}}, \tag{8}$$

where n is the number of sampled states. Each feature vector possesses a dimension of $3n \times 1$.

The arcs sampled from the same fundamental solution are clustered by their geometry. First, the arcs are coarsely clustered in the shape-based feature vector space using HDBSCAN.[25,27] The governing parameters are selected as $m_{pts} = 4$, $m_{minclust} = 5$, and $\epsilon_{merge} = 2\sin(5°/2)$.[33] Each coarse group is then refined using a process developed by Bosanac that is modeled after convoy detection to define each cluster as a set of arcs with a consistently similar shape and path.[33,34] First, the ith sampled state along each arc in the coarse group is described by $\boldsymbol{f}_s$ and $\boldsymbol{f}_p$, producing two three-dimensional feature vectors. These sampled states are clustered in each feature vector space independently using DBSCAN.[24] The governing parameters are selected as $m_{pts} = 4$ and $\epsilon = (m_{pts} + 1)\max(e_{maxk}, \epsilon_{thresh})$, where e_{maxk} is the $m_{minclust}$-largest distance from each member to its nearest neighbor and ϵ_{thresh} is a specified threshold, equal to $2\sin(5°/2)$ in $\boldsymbol{f}_s$ and 10^{-3} in $\boldsymbol{f}_p$.[33] This procedure is repeated for all n states along the arc to produce $2n$ clustering results. Any arcs that are consistently grouped together in all clustering results produce a refined cluster. This step produces a set of clusters $\mathcal{L}$, localized to each half-manifold of each periodic orbit. Similar motion types that exist across different families of fundamental solutions are aggregated together using pairwise merging decisions: 1) nearest neighbor representatives are calculated in each of the position and velocity feature vector spaces, 2) the arcs in each pair of clusters are input to the cluster refinement process, and 3) two clusters with members that are grouped together are merged.[16,33] This aggregation process results in a set of global clusters $\mathcal{G}_j$ summarizing arcs that approach or depart members of the jth family of periodic orbits.

Periodic orbits are discretely sampled along continuous families and their curvature profiles are analyzed to extract motion primitives. This procedure was previously presented by Gillespie, Miceli, and Bosanac.[16] Groups of geometrically similar orbits within each family are created by calculating the number of curvature maxima along each family. Then, sets of neighboring members along the family with the same number of curvature maxima form a geometrically similar group.

To support trajectory prediction, each cluster is summarized by its motion primitive and region of existence.[16] First, the motion primitive is selected as the arc that is equal to the medoid of the cluster, evaluated in the position-based feature vector space. Next, the associated region of existence, i.e., the volume that the cluster encompasses within the phase space, is approximated through a discrete voxel representation. To support accurate bounding of these regions, the continuous trajectories are further sampled in even increments of 0.05 non-dimensional arclength and velocity arclength to produce a finer discrete set of states. The fine resolution of these samples is shown in Figure 2(b), where the region of existence of a set of geometrically similar trajectories that departs an L_1 Lyapunov orbit is plotted around the trajectory samples. The voxels are colored by the highest section index of any intersecting samples, the samples for a subset of the trajectories are shown in blue, and the primitive trajectory is overlaid in black. Additionally, in Figure 2(b), trajectories begin within section 1, the red diamonds locate L_1 and L_2, and the gray sphere represents the Moon (to scale).

Each voxel signifies a small region in the configuration space that contains at least one state along a trajectory within the cluster associated with the selected primitive. These voxels also contain data encoding the velocity of states that pass through the voxel. The centers of the voxels themselves act as rounded position values. In a similar manner, the encoded velocities are rounded to further reduce the data size of each region of existence object. The voxels inside each object also contain additional information, including an index for each trajectory of the cluster that passes through it and a list of section indices.

This paper leverages a library of 6,861 motion primitives. This library includes arcs that follow, approach, or depart selected L_1 and L_2 Lyapunov orbits along with the Moon-centered distant prograde orbits, distant retrograde orbits, and low prograde orbits. Future work will incorporate spatial and thrusting primitives as well.

Estimation Mapping

An uncertain state estimate is projected onto the motion primitive library to identify short-term motion types that it may follow. First, a three-sigma uncertainty ellipsoid centered at an estimate mean $\bar{\boldsymbol{x}}$ is defined analytically in the phase space. Then, the velocities contained within the N voxels of a primitive's region of existence are combined with their respective voxel center positions to form a list of rounded states $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, N$. A series of intersection checks are performed to decide if a corresponding full precision state could have intersected the uncertainty ellipsoid. A preliminary coarse intersection check is performed to exclude $\boldsymbol{x}_i$ that are far from the estimate mean. For each $\boldsymbol{x}_i$, consider a six-dimensional hypercube centered at $\boldsymbol{x}_i$ with side length l_i equal to the larger of the position voxel side length and the velocity discretization step. This hypercube is subsequently converted to a hypersphere with center $\boldsymbol{x}_i$ and radius equal to the distance from the center of the hypercube to one of its corners, calculated as $d_i = (\sqrt{6}/2)l_i$. This hypersphere serves as a conservative bound on the corresponding full precision state. Additionally, the uncertainty ellipsoid is approximated as a hypersphere centered at $\bar{\boldsymbol{x}}$ with radius equal to the largest principal axis length a_{max} of the three-sigma uncertainty ellipsoid. To identify $\boldsymbol{x}_i$ whose full precision counterparts cannot possibly intersect the uncertainty ellipsoid, any $\boldsymbol{x}_i$ meeting the condition

$$\| \boldsymbol{x}_i - \bar{\boldsymbol{x}} \| > a_{max} + d_i \tag{9}$$

are removed from consideration. This quick operation greatly reduces the number of necessary precise intersection checks.

As an intermediate precision check, to determine whether a rounded state itself lies within the three-sigma error ellipsoid, the squared Mahalanobis distance[35] of $\boldsymbol{x}_i$ from the mean state estimate is computed and used to evaluate the inequality

$$(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) \; - \; \sigma^2 \; \leq \; 0, \tag{10}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix defining the ellipsoid and $\sigma = 3$ corresponds to the three-sigma confidence level. If this inequality holds, $\boldsymbol{x}_i$ lies inside the uncertainty ellipsoid and is marked as intersecting; otherwise, a constrained root-finding method is employed to check if a corresponding full precision state could intersect the uncertainty ellipsoid. The root finding method works by solving for a full precision state positioned within the same voxel, that causes the left-hand-side of Equation 10 to evaluate to zero subject to linear inequality constraints. These constraints force the position component to lie within the voxel position bounds and the velocity component to lie within plus or minus the velocity discretization step from the rounded state's velocity. If a solution is found, then the rounded state is conservatively marked as intersecting. Any rounded state found to intersect the uncertainty ellipsoid is stored, along with information such as the intersecting voxel, section, and trajectory. This process is repeated for each primitive in the library to produce a list of primitives that intersect the uncertainty ellipsoid. These primitives supply a digestible summary of possible short-term trajectory predictions.

In this paper, an example observation is generated through the random sampling of trajectories. A trajectory that departs an L_1 Lyapunov orbit is propagated one thousand times for a duration of one day with a specified standard deviation in initial position and velocity. For reference, when processing ground tracking measurements from the THEMIS-ARTEMIS mission, Woodard et al. found that position and velocity accuracies of 20 m and 0.002 cm/s, respectively, were achievable for spacecraft in libration point orbits with ten days of batch processing.[36] To approximate a reasonable position uncertainty near a libration point orbit, a standard deviation of 20 m is selected; To simulate a thruster misfire or similar event, a larger velocity standard deviation of 10 m/s is used. After propagating, a mean state estimate $\bar{\boldsymbol{x}}$ is extracted from the terminal states along with a covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^6$. These trajectories are visualized in Figure 3 in black with $\bar{\boldsymbol{x}}$ as a blue star. The resulting three-sigma uncertainty ellipsoid is projected onto $\mathbb{R}^3$ and plotted in transparent red, possessing a similar size to the voxels utilized in this work, 0.01 nondimensional position units.

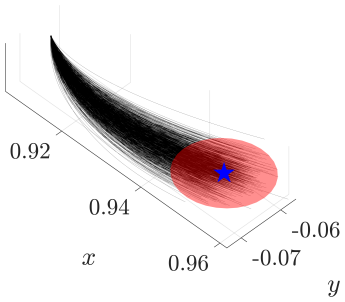
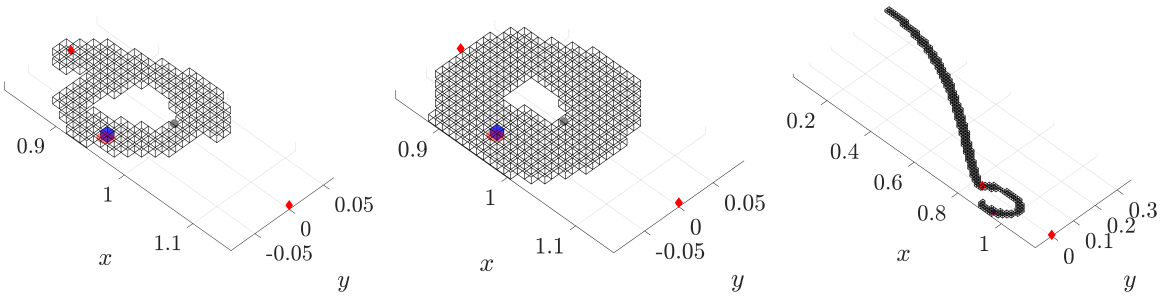


Figure 3: Trajectories propagated to generate an uncertainty ellipsoid (projected onto $\mathbb{R}^3$) and $\bar{\boldsymbol{x}}$.

Mapping the state estimate depicted in Figure 3 against the library of 6,861 motion primitives identifies a set of 18 primitives in approximately three minutes. In this paper, all computations are performed using a 6-core AMD Ryzen 5 7600X CPU. The majority of computations are highly parallelizable and scale well on clusters with higher core counts. Three of these primitives are visualized in Figure 4, where intersecting voxels are highlighted blue and the uncertainty ellipsoid projected onto $\mathbb{R}^3$ is in red. The regions of existence all possess similar shapes nearby the initial state estimate, but the subsequent geometries and itineraries vary drastically: Figure 4(a) displays a self-intersecting geometry that gradually moves towards the L_1 gateway, Figure 4(b) depicts a different self-intersecting geometry that remains within the vicinity of the Moon, and Figure 4(c) shows a third possible geometry that quickly departs through the L_1 gateway. Thus, the selected uncertain state estimate could potentially lead to a spacecraft following a variety of distinct itineraries.



(a) (b) (c)

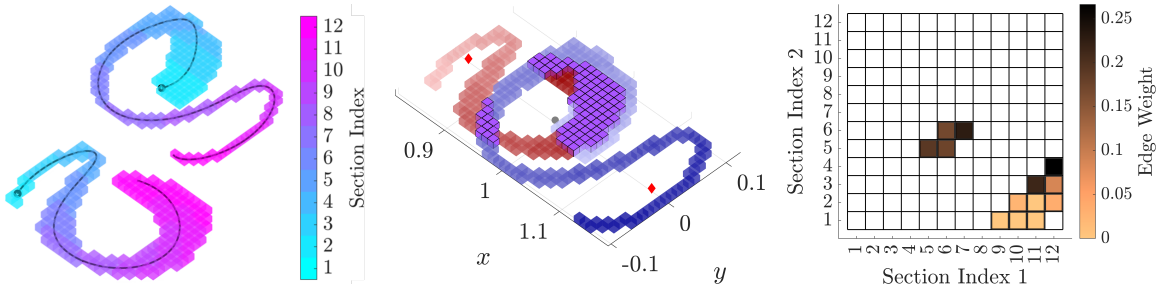
Figure 4: Three regions of existence that intersect with the uncertainty ellipsoid in Figure 3.

Motion Primitive Graph Construction

To support long-term trajectory prediction, sequences of composable primitives are generated. To summarize the potential sequential composability of behavioral motion primitives from the library, a high-level graph is constructed. This step leverages and builds upon prior work by Miceli and Bosanac[19] to increase the accuracy by leveraging the voxel representation of each region of existence. To demonstrate this process, consider the regions of existence of two primitives, one containing trajectories that departs an L_1 Lyapunov orbit and another containing trajectories approaching an L_2 Lyapunov orbit, as plotted in Figure 5(a). In this subfigure, the color varies along each region of existence from cyan at the initial section to magenta at the final section.

The graph is first populated with nodes that are defined using the motion primitives in the library. Each node represents a single section along the region of existence of a motion primitive. Thus, a region of existence containing 12 sections contributes 12 nodes to the graph.

Zero-weight, directed edges connect sequential sections associated with the same motion primitive. Accordingly, these edges reflect that a spacecraft can naturally traverse the continuous sections of trajectories with a similar geometry. The final section node for periodic orbits is connected with a similar, zero-weight edge to the first node to encode periodicity.



(a) Departing an L_1 Lyapunov orbit and approaching an L_2 Lyapunov orbit

(b) The two regions colored red and blue, with their intersections colored purple

(c) Edge weights between intersecting sections of the two regions

Figure 5: Identifying intersections between the regions of existence associated with a pair of behavioral motion primitives to estimate their potential for sequential composability.

Directed edges are added between selected nodes associated with intersecting motion primitives that are considered sequentially composable. To identify intersections between each pair of motion primitives in the library, the center of every voxel from one primitive's region of existence is compared to the centers of the voxels from the other primitive's region of existence. If two voxel centers overlap, those voxels and the corresponding sections in each region are marked as intersecting. Figure 5(b) represents each region of existence in either red or blue with the colors darkening as the section index increases; the intersecting voxels are then colored in purple. For each unique combination of section indices along both regions of existence, two directed edges are added between the associated nodes in the graph. This bidirectionality reflects that initial guesses could be constructed using these two sections sequentially composed in either order.

To calculate the weight of the edge connecting section a of one primitive and section b of the other primitive, let $\mathcal{V}_a$ and $\mathcal{V}_b$ denote the sets of velocities in all intersecting voxels within sections a and b, respectively. Then, the edge weight between the two sections $w_{a,b}$ is defined as the minimum normalized velocity discontinuity over all pairs $(\boldsymbol{v}_i, \boldsymbol{v}_j)$ in their respective sets

$$w_{a,b} = \min_{\substack{\boldsymbol{v}_i \in \mathcal{V}_a \\ \boldsymbol{v}_j \in \mathcal{V}_b}} \frac{\|\boldsymbol{v}_i - \boldsymbol{v}_j\|}{\|\boldsymbol{v}_i\| + \|\boldsymbol{v}_j\|}. \tag{11}$$

Normalizing the edge weight by the speeds focuses on analyzing relative changes in the velocity, which is useful when generating sequences of primitives that may pass close to the Moon with higher speeds.[19] Figure 5(c) displays the calculated edge weight of each section pair from the two selected primitives, labeling the red and blue voxel sets from Figure 5(b) as regions of existence 1 and 2, respectively. Two groups of intersecting sections are visible in the grid: a smaller, higher edge weight group and a larger, lower edge weight group. These groupings are consistent with Figure 5(b) where there is a large purple voxel intersection region with well-aligned velocity vectors and a smaller purple voxel intersection region with larger differences in the velocity directions. For inter-primitive edges with a calculated edge weight of zero ($\boldsymbol{v}_i = \boldsymbol{v}_j$), the actual edge weight is set to 10^{-14}. This value captures that $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are rounded quantities and any trajectory connecting them would likely require an impulsive maneuver of nonzero magnitude. This also helps the graph search prioritize flow edges over inter-primitive edges when appropriate.

As previously demonstrated by Miceli and Bosanac,[19] additional path or maneuver constraints can be applied when calculating edge weights. In this paper, a maximum velocity angle difference of 30° is applied except for velocity pairs where both velocity norms are below 10 m/s. This constraint avoids populating the graph with sharp turns and unnecessarily high maneuver costs.

Generating Motion Primitive Sequences

The motion primitive graph is searched to generate motion primitive sequences that support generating long-term trajectory predictions from an uncertain state estimate. The estimation mapping process provides a list of start nodes, corresponding to sections along primitives that overlap with the uncertainty ellipsoid. Sequences of motion primitives that begin from these nodes support generating a summary of reachable orbits, regions, or vantage points. In this proof of concept, a set of candidate destinations are selected as libration point and Moon-centered periodic orbits, the Moon's surface, and the L_1 and L_2 gateways. A primitive sequence that connects the initial uncertainty ellipsoid to any of these destinations is used to generate a continuous trajectory in the next step and, therefore, supply a single long-term trajectory predication. Furthermore, the collection of primitive

sequences supplies geometrically distinct pathways to reachable destinations, potentially supplying a digestible set of long-term trajectory predictions.

This paper uses a K-shortest paths search algorithm developed by Miceli and Bosanac[19] and based on Yen's algorithm to generate multiple paths through the graph between selected start and end nodes. These paths correspond to sequences of nodes and edges that are transformed to primitive sequences. The only difference from the standard Yen's algorithm, is that rather than removing the single edge from a spur node to the subsequent node in the original path, all edges from the spur primitive to the next primitive in the sequence are removed. Within the modified Yen's algorithm, Dijkstra's algorithm is used to compute each candidate path with MATLAB's *shortestpath*.

The target node is specified before each graph search. If the destination is a periodic orbit group, its associated region of existence is located. Then, one node corresponding to a section along the primitive is selected. The nodes of the periodic orbit group in the graph can be freely traversed in a loop, so the search algorithm will naturally choose the optimal node to insert onto. If, however, the destination is the Moon's surface, then a ring of voxels each with a side length of 1 km is placed at the Moon's surface to target a close approach to the Moon with an altitude of less than approximately 500 m. This ring of voxels is represented as a single node in the graph and marked as the target node. Finally, if the destination is the L_1 or L_2 gateway, then a line of voxels is placed in the $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ plane and parallel to the $\hat{\boldsymbol{y}}$ axis at $x = 0.75$ or $x = 1.23$, respectively. These lines of voxels span $y = [-0.125, 0.125]$ to target a variety of escape geometries and are each assigned a single node in the graph. The corresponding node is selected as the target node.

For each of the 18 initial primitives identified by the estimation mapping process, 50 paths are generated to each of the 22 destinations by searching the motion primitive graph. This search process results in 19,800 paths that require approximately two hours to compute. Two of these primitive subsequences are plotted in Figure 6. In each subfigure, the series of sections selected by the graph search are plotted using portions of the respective regions of existence. Each section is colored according to its primitive. The first primitive in the sequence, which intersected the observation uncertainty ellipsoid, has its sections colored purple. In Figure 6(a), the path arrives at an L_1 Lyapunov orbit primitive. In Figure 6(b), the path escapes through the L_2 gateway. Both motion primitive sequences intersect relatively smoothly, increasing the likelihood of producing a good initial guess for a trajectory with impulsive maneuvers.

Generating an Initial Guess from a Primitive Sequence

To generate initial guesses, the sequences of primitive sections are processed. This refinement process is implemented in a manner similar to Miceli and Bosanac,[19] through the construction of a low-level graph that uses states along an array of trajectories from each region of existence for nodes. A representative subset of the trajectories that lie within the region of existence of each primitive in a sequence is identified by clustering in the position-based feature vectors space using the k-medoids algorithm, accessed in MATLAB. This clustering step samples the set of trajectories associated with each primitive to reduce the low-level graph size and computational costs. Then, the sampled states along each selected trajectory define the nodes in the low-level graph.

Edges are added between nodes associated with states along the same trajectory and between composable states from sequential primitives. Zero-weight, directed edges are added between nodes defined using sequential states sampled from the same trajectory, ensuring that trajectories can be traversed without penalty. Similar zero-weight edges are added from the last node to the first node

along trajectories following periodic orbits to encode periodicity. Directed, weighted edges are then added between nodes on distinct primitives that are sequentially composable. These edge weights W between the nodes are calculated by weighting position and velocity discontinuities as

$$W = w_p \left|\Delta r\right| + w_v(1 - \cos\theta), \tag{12}$$

where Δr is the position difference between two corresponding states and θ is the angle between their velocity vectors. Scalar coefficients $w_p = 10$ and $w_v = 1$ are used to balance the relative importance of connections with close proximity or well aligned velocity directions, respectively.

Edges are added between trajectories of each primitive set according to the high-level graph. As a result, at each node of the graph the only choices are to continue along the current trajectory or jump to a state along a trajectory associated with the next primitive in the higher level path.

The modified A* search algorithm, developed by Miceli and Bosanac,[19] is used to search this graph for a sequence of nodes and edges that minimizes the cumulative edge weight. One modification requires the path consist of at least two nodes along any trajectory, i.e., every arc must be traversed for a nonzero duration. A list of potential start nodes are selected from the trajectories of the primitive set that intersect the uncertainty ellipsoid. From this set, one potential start node per trajectory is selected as the state closest to the mean estimate. The list of multiple start nodes is incorporated by adding them all to the priority queue when initializing the modified A* algorithm. A list of potential end nodes is populated from the final nodes along each primitive in the destination set. Accordingly, an additional modification to A* is the allowance of multiple end nodes by 1) allowing termination upon reaching any end node and 2) setting an A* heuristic $h(n)$ that considers all possible end nodes. The selected A* heuristic is the cost of the shortest path from the current node n to any end node. This cost is calculated without considering the two node sub-path constraint and will always be less than or equal to the true cost when accounting for the constraint to satisfy the admissible heuristic criteria for A*.

Of the 19,800 high-level paths generated, 18,252 of them are successfully refined to produce initial guesses in under two hours. A failure to refine indicates that the A* algorithm failed to find a viable path through the low-level graph. Two of the refined initial guesses are included in Figure 7 plotted with dashed lines. Each unique color in the initial guess corresponds to a different segment

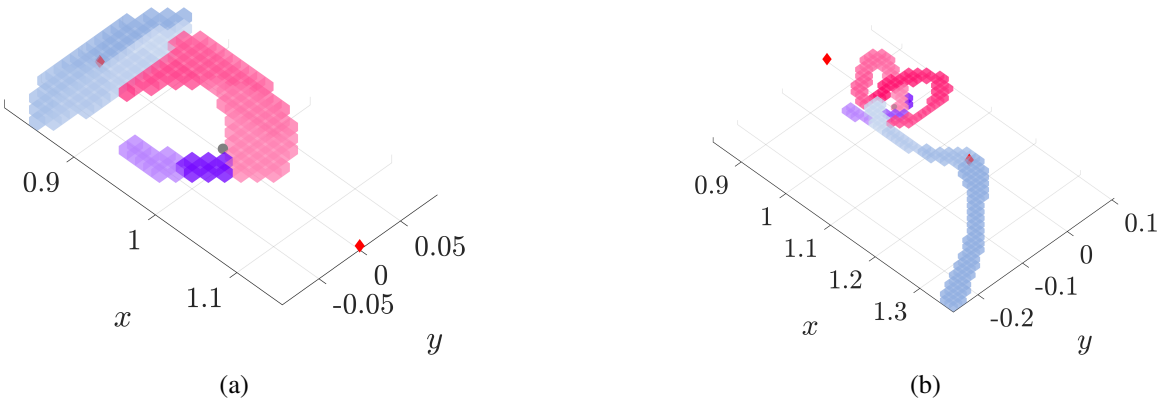
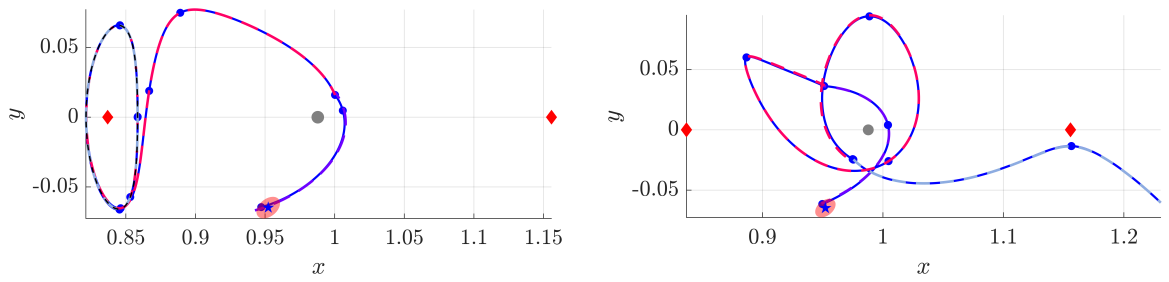


Figure 6: Two primitive subsequences from the observation to (a) an L_1 Lyapunov orbit primitive and (b) L_2 gateway escape.



(a) (b)

Figure 7: Two discontinuous initial guesses and the corrected trajectories: (a) from the observation to an L_1 Lyapunov orbit and (b) from the observation to an escape through the L_2 gateway.

of a primitive from the high-level graph search. These initial guesses exhibit a wide variety of geometries with small position discontinuities and velocity direction changes.

Correcting Initial Guesses Using Multiple Shooting

Each discontinuous initial guess is corrected via multiple-shooting to produce a continuous impulsive maneuver-enabled trajectory. Initial guesses for impulsive maneuvers are placed at maxima in the curvature and at the boundaries of trajectories from distinct primitives. At these points, only position constraints are enforced. Two samples are evenly distributed in the arclength between each of these impulsive maneuvers, and full state continuity is enforced between the associated arcs. In addition, the initial position is constrained to lie within the three-sigma uncertainty ellipsoid in configuration space. This constraint is enforced through the addition of a slack variable β_e. The final position is constrained to match the final position of the initial guess, $\boldsymbol{x}_{f,IG}$. Further constraints are added to ensure that periapses maintain an altitude of 10 km above the Moon's surface. Two constraints are added for each of the $j = 1 \dots m$ identified periapsis events

$$\boldsymbol{F}_{p,j} = \begin{bmatrix} (\boldsymbol{r}_{p,j} - \boldsymbol{r}_{moon}) \cdot \boldsymbol{v} \\ \left\| \boldsymbol{r}_{p,j} - \boldsymbol{r}_{moon} \right\| - r_{moon,rad} - \beta_{p_j}^2 \end{bmatrix}. \tag{13}$$

This implementation assumes that for a good initial guess, a periapsis event will not become an apoapsis event and, therefore, the sign of the time derivative of the first constraint is not incorporated into $\boldsymbol{F}_{p,j}$. Finally, constraints are added to enforce positive time of flight along each arc using slack variables β_{t_i} for arcs $i = 1 \dots n$. The set of additional constraints $\boldsymbol{F}_{add}$ is defined as

$$\boldsymbol{F}_{add} = \begin{bmatrix} (\boldsymbol{x}_{1,0} - \bar{\boldsymbol{x}})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_{1,0} - \bar{\boldsymbol{x}}) - \sigma^2 + \beta_e^2 \\ \boldsymbol{x}_{n,f} - \boldsymbol{x}_{f,IG} \\ \boldsymbol{F}_{p,1} \\ \vdots \\ \boldsymbol{F}_{p,j} \\ \Delta T_1 - \beta_{t_1}^2 = 0 \\ \vdots \\ \Delta T_n - \beta_{t_n}^2 = 0 \end{bmatrix} \tag{14}$$

Together, all the constraints are used to define the full constraint vector as $\boldsymbol{F} = [\boldsymbol{F}_0^{\mathrm{T}} \ \boldsymbol{F}_{add}^{\mathrm{T}}]^{\mathrm{T}}$.

The initial guesses, derived from primitive sequences, are corrected. During the corrections process, the constraint vector must reach a norm of less than or equal to $\tau = 10^{-10}$ within 100 iterations. Of the 18,252 initial guesses generated, 14,254 of them are successfully corrected to produce continuous trajectories in about twenty minutes. Two corrected trajectories are included in Figure 7, overlaid with their corresponding initial guesses and marked with circles to indicate impulsive maneuver locations. The corrected trajectories retain the geometry of their initial guesses, but are now continuous at each arc intersection. Optimizing these trajectories to reduce their total maneuver magnitudes is an avenue of ongoing work.

Generating a Digestible Trajectory Prediction Summary

Generating a digestible summary of possible trajectories from an observation requires a reduction in the total number of trajectories. To reduce the number of trajectories included in the summary, their similarity is assessed through the use of dynamic time warping (DTW).[37] DTW is typically used in time series analysis for comparing signals of varying speed.[38] This distance measure pairs similar samples from two sets to minimize the sum of Euclidean distances between the pairs. A sample from one set may be paired with multiple, sequential samples from the other set if necessary to produce the minimum sum of distances.

Before applying DTW to the corrected trajectories, they are grouped by destination and sorted by their total maneuver magnitude within the groups. The trajectory with the minimum total maneuver magnitude from each group is added to the group's final list. Each subsequent trajectory is then compared to the trajectories in the list through two separate applications of dynamic time warping, in position and velocity, to determine if it is sufficiently different to be added to the final list. If the position and velocity similarity metrics from DTW are both above a manually selected similarity threshold, the trajectory is added to the list for its respective group. The similarity thresholds are tuned for the individual destinations and range from 1-2 and 2-10 nondimensional units for position and velocity, respectively.

RESULTS AND DISCUSSION

Trajectory predictions to a variety of destinations are generated and corrected using the process detailed in this paper. Table 1 summarizes the ranges of total maneuver costs, flight time (TOF), and final Jacobi constants for each prediction set. The destination orbit period is also included when relevant. Total maneuver magnitudes (before optimization) vary from approximately 0.02 km/s to several km/s, flight durations span from about 0.7 days ($\approx$17 hours) to 84 days, and final Jacobi constants range between 2.63 and 3.19. For reference, the Jacobi constant of the observation estimate mean $\bar{\boldsymbol{x}}$ is 3.19.

The predicted trajectories are plotted in Figures 8 and 9, respectively. Each trajectory is plotted in a distinct color, final periodic orbits are drawn with black dashed lines, circle markers indicate impulsive maneuver locations, and the libration points are depicted with red diamonds. For particularly extensive paths, adjacent zoomed figures provide additional detail.

The generated summary includes long-term trajectory predictions with a wide variety of geometries, including multiple lunar revolutions, direct gateway escapes, and Moon impacts. Traditional methods such as Poincaré mapping[39] and Monte Carlo[4] sampling, while effective for many situations, can be time-consuming and produce an overwhelming volume of data, rendering them impractical for rapid decision-making. Poincaré maps require extensive computation and manip-

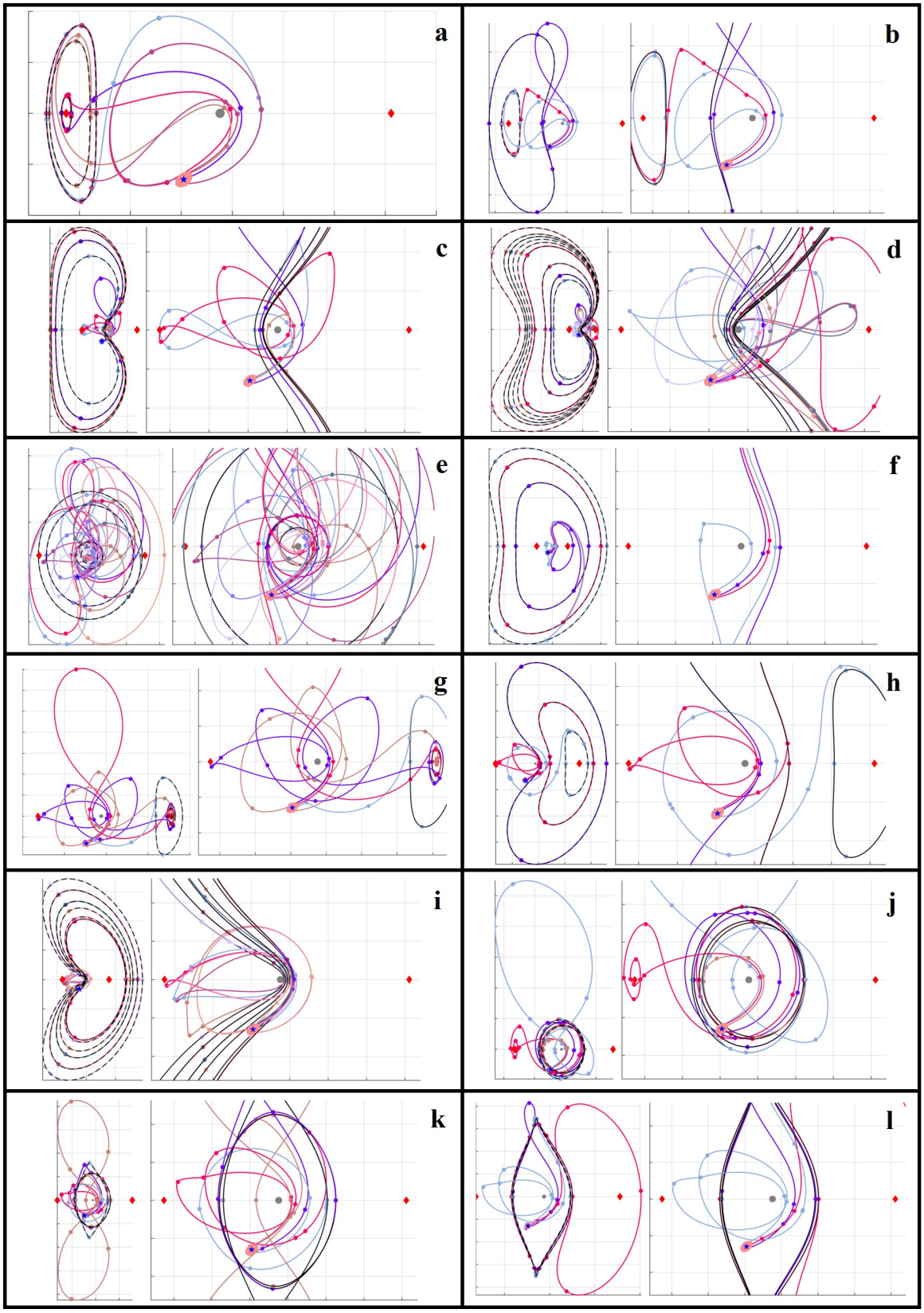


Figure 8: Trajectory predictions for destinations a through l

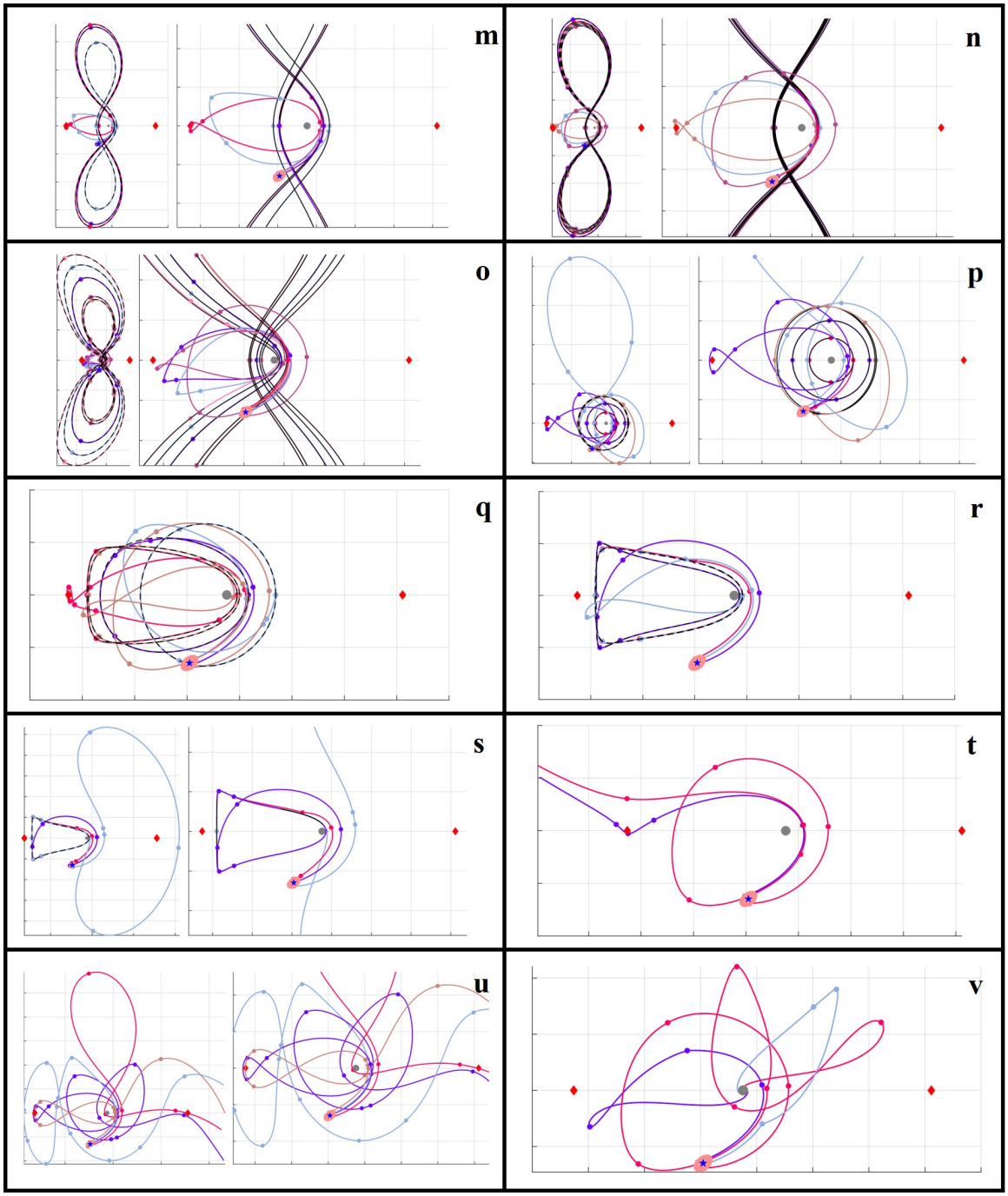


Figure 9: Trajectory predictions for destinations m through v

Table 1: Ranges of Δv (km/s), flight time (days), final Jacobi constant, and period of destination periodic orbit (where applicable) for each destination in Figures 8 and 9.

Set	L_1 Lyapunov				Distant Retrograde	
	a	b	c	d	e	f
ΔV	[0.05, 0.61]	[0.13, 0.65]	[0.27, 1.19]	[0.34, 1.70]	[0.35, 3.74]	[0.43, 1.57]
TOF	[22.2, 32.3]	[30.1, 36.8]	[29.3, 49.6]	[32.9, 54.4]	[7.3, 38.1]	[41.9, 54.6]
$C_{J,f}$	[3.16, 3.19]	[3.02, 3.15]	[2.91, 2.95]	[2.52, 2.91]	[2.95, 3.40]	[2.63, 2.82]
P	[11.7, 12.2]	[12.3, 17.1]	[23.8, 28.7]	[28.3, 32.4]	[0.9, 11.4]	[23.9, 26.5]

Set	L_2 Lyapunov			Distant Prograde		
	g	h	i	j	k	l
ΔV	[0.14, 0.68]	[0.12, 0.28]	[0.17, 0.49]	[0.18, 0.75]	[0.15, 0.64]	[0.13, 0.24]
TOF	[18.9, 44.7]	[23.8, 36.1]	[24.9, 44.6]	[6.9, 41.6]	[16.7, 42.7]	[23.9, 44.5]
$C_{J,f}$	[3.15, 3.17]	[2.97, 3.13]	[2.90, 2.96]	[3.16, 3.18]	[3.10, 3.14]	[3.09, 3.10]
P	[14.7, 14.9]	[15.1, 23.3]	[24.4, 32.4]	[5.9, 8.1]	[9.9, 13.7]	[14.0, 15.0]

Set	Distant Prograde			Low Prograde		
	m	n	o	p	q	r
ΔV	[0.12, 0.27]	[0.14, 0.29]	[0.17, 0.45]	[0.13, 0.60]	[0.06, 0.45]	[0.09, 0.47]
TOF	[26.8, 54.3]	[28.8, 62.7]	[29.2, 83.7]	[3.2, 27.6]	[9.1, 31.7]	[11.8, 31.3]
$C_{J,f}$	[3.00, 3.02]	[2.99, 3.00]	[2.94, 2.99]	[3.19, 3.42]	[3.18, 3.19]	[3.18, 3.18]
P	[23.1, 26.7]	[27.0, 28.3]	[28.9, 40.5]	[1.2, 5.1]	[5.3, 10.3]	[10.9, 11.8]

Set	Low Prograde	L_1 Gateway	L_2 Gateway	Moon Impact
	s	t	u	v
ΔV	[0.08, 1.13]	[0.02, 0.17]	[0.10, 0.86]	[0.22, 0.93]
TOF	[12.5, 41.6]	[12.5, 12.8]	[14.1, 33.6]	[5.4, 17.4]
$C_{J,f}$	[3.18, 3.18]	[3.16, 3.19]	[3.02, 3.16]	[3.17, 3.19]
P	[11.8, 11.8]	—	—	—

ulation to identify reachable sets, whereas Monte Carlo approaches lack a natural mechanism to incorporate intentional maneuvers or destination constraints. In contrast, the behavioral motion primitive framework delivers concise, destination-focused summaries. The periodic-orbit prediction sets delineate accessible vantage points suitable for long-duration missions. The L_1 and L_2 gateway escape prediction sets highlight available departure options or warn of inadvertent exits from the lunar region. Similarly, the lunar impact trajectory set informs risk assessments or planned impact strategies, depending on mission objectives.

Due to the lack of a maximum maneuver constraint in the high-level graph search, some high maneuver cost trajectories are returned. These may not be reachable for every spacecraft, but still contribute important information to the prediction summary. However, optimization is expected to further reduce the total maneuver magnitudes in future work.

CONCLUSION

This paper presents a new approach to long-term spacecraft trajectory prediction in cislunar space by leveraging a library of motion primitives. Each primitive's region of existence is approximated via discrete voxels with encoded velocity information, enabling rapid mapping of uncertain state estimates onto a set of short-term motion predictions. By constructing and searching a motion primitive graph, sequences of composable primitives are generated from the initial state estimate to various vantage points. Diverse trajectory prediction sets are generated in the CR3BP for destinations including 18 periodic orbit primitives, L_1 and L_2 gateway escapes, and lunar impacts. Future work will extend this framework to spatial and continuous thrust primitives.

ACKNOWLEDGMENT

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-23-1-0235. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force.

REFERENCES

[1] P. R. Escobal, Methods of Orbit Determination, pp. 239–292. Malabar, FL: Krieger, 1965.

[2] R. H. Gooding, "A New Procedure for Orbit Determination Based on Three Lines of Sight (Angles Only)," Tech. Rep. TR-93004, Defense Research Agency, Farnborough, Hampshire, England, 1993.

[3] B. D. Tapley, B. E. Schutz, and G. H. Born, Statistical Orbit Determination, pp. 194–199. New York: Elsevier, 2004.

[4] Y. z. Luo and Z. Yang, "A review of uncertainty propagation in orbital mechanics," Progress in Aerospace Sciences, Vol. 89, 2017, pp. 23–39, https://doi.org/10.1016/j.paerosci.2016.12.002.

[5] C. Frueh, K. Howell, K. J. Demars, and S. Bhadauria, "Cislunar Space Situational Awareness," 2021.

[6] H. Sorenson and D. Alspach, "Recursive bayesian estimation using gaussian sums," Automatica, Vol. 7, No. 4, 1971, pp. 465–479, https://doi.org/10.1016/0005-1098(71)90097-5.

[7] A. Wolek and C. A. Woolsey, Model-Based Path Planning. Cham: Springer International Publishing, 2017, 10.1007/978-3-319-55372-6_9.

[8] O. Dermy, A. Paraschos, M. Ewerton, J. Peters, F. Charpillet, and S. Ivaldi, "Prediction of Intention during Interaction with iCub with Probabilistic Movement Primitives," Frontiers in Robotics and AI, Vol. 4, 2017, 10.3389/frobt.2017.00045.

[9] D. Clever, M. Harant, H. Koch, K. Mombaur, and D. Endres, "A Novel Approach for the Generation of Complex Humanoid Walking Sequences Based on a Combination of Optimal Control and Learning of Movement Primitives," Robotics and Autonomous Systems, Vol. 83, 2016, pp. 287–298, 10.1016/j.robot.2016.06.001.

[10] S. Ferguson, B. Luders, R. C. Grande, and J. P. How, "Real-Time Predictive Modeling and Robust Avoidance of Pedestrians with Uncertain, Changing Intentions," CoRR, Vol. abs/1405.5581, 2014.

[11] A. Majumdar and R. Tedrake, "Funnel Libraries for Real-Time Robust Feedback Motion Planning," CoRR, Vol. abs/1601.04037, 2016.

[12] G. Habibi, N. Jaipuria, and J. P. How, "SILA: An Incremental Learning Approach for Pedestrian Trajectory Prediction," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 4411–4421, 10.1109/CVPRW50498.2020.00520.

[13] H. Edelhoff, J. Signer, and N. Balkenhol, "Path segmentation for beginners: an overview of current methods for detecting changes in animal movement patterns," Movement Ecology, Vol. 4, Sep 2016, p. 21, 10.1186/s40462-016-0086-5.

[14] T. R. Smith and N. Bosanac, "Motion Primitive Approach to Spacecraft Trajectory Design in a Multibody System," The Journal of the Astronautical Sciences, Vol. 70, No. 34, 2023, 10.1007/s40295-023-00395-7.

[15] N. Bosanac, "Clustering Natural Trajectories in the Earth-Moon Circular Restricted Three-Body Problem," 2025, Under Review.

[16] C. Gillespie, G. E. Miceli, and N. Bosanac, "Summarizing Natural and Controlled Motion in Cislunar Space With Behavioral Motion Primitives," AAS/AIAA Space Flight Mechanics Meeting, 2025.

[17] G. Miceli, N. Bosanac, and R. Karimi, "Generating The Trajectory Design Space For Neptunian System Exploration," AAS/AIAA Astrodynamics Specialists Conference, August 2024.

[18] G. E. Miceli and N. Bosanac, "Designing Neptunian System Tours via a Motion Primitive Approach," AAS/AIAA Space Flight Mechanics Meeting, 2025.

[19] G. E. Miceli and N. Bosanac, "Generating the trajectory design space for Neptunian system exploration," 2025, Under Review.

[20] V. Szebehely, Theory of Orbits: The Restricted Problem of Three Bodies. London: Academic Press, 1967.

[21] K. Wardle, Differential Geometry. Mineola, NY: Dover Publications, Inc., 2008.

[22] N. Patrikalakis, T. Maekawa, and W. Cho, Shape Interrogation for Computer Aided Design and Manufacturing. Springer, Berlin, Heidelberg, 2009. E-book.

[23] N. Bosanac, "Curvature Extrema Along Trajectories in the Circular Restricted Three-Body Problem," AAS/AIAA Astrodynamics Specialists Conference, 2024.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, AAAI Press, 1996, p. 226–231.

[25] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," Advances in Knowledge Discovery and Data Mining (J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, eds.), Berlin, Heidelberg, Springer Berlin Heidelberg, 2013, pp. 160–172.

[26] C. Malzer and M. Baum, "A Hybrid Approach To Hierarchical Density-based Cluster Selection," 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), 2019, pp. 223–228.

[27] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical Density Based Clustering," The Journal of Open Source Software, Vol. 2, mar 2017, 10.21105/joss.00205.

[28] A. Kaufman, D. Cohen, and R. Yagel, "Volume graphics," Computer, Vol. 26, No. 7, 1993, pp. 51–64, 10.1109/MC.1993.274942.

[29] W. Dijkstra, E. "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, Vol. 1, Dec 1959, pp. 269–271, 10.1007/BF01386390.

[30] E. Hart, P. J. Nilsson, N. and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, Vol. 4, No. 2, 1968, pp. 100–107, 10.1109/TSSC.1968.300136.

[31] J. Y. Yen, "An Algorithm for Finding Shortest Routes from All Source Nodes to a Given Destination in General Networks," Quarterly of Applied Mathematics, Vol. 27, 1970, pp. 526–530.

[32] J. S. Parker and R. L. Anderson, Low-energy Lunar Trajectory Design. Hoboken, New Jersey: Wiley, 2014.

[33] N. Bosanac, "Data-Driven Summary of Motion in an Ephemeris Model of Cislunar Space," AAS/AIAA Space Flight Mechanics Meeting, Lihue, HI, 2025.

[34] Y. Zheng and X. Zhou, Computing with Spatial Trajectories. Springer Publishing Company, Incorporated, 1st ed., 2011, 10.1007/978-1-4614-1629-6.

[35] P. C. Mahalanobis, "On the generalized distance in statistics," Proceedings of the National Institute of Sciences of India, Vol. 2, 1936, pp. 49–55.

[36] M. Woodard, D. Cosgrove, P. Morinelli, J. Marchese, B. Owens, and D. Folta, "Orbit Determination of Spacecraft in Earth–Moon L1 and L2 Libration Point Orbits," Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Girdwood, AK, USA, American Astronomical Society and American Institute of Aeronautics and Astronautics, July 2011. Paper AAS 11-515, GSFC.CP.4811.2011.

[37] K. L. Bruchko and N. Bosanac, "Rapid Trajectory Design in Multi-Body Systems Using Sampling-Based Kinodynamic Planning," The Journal of the Astronautical Sciences, Vol. 72, July 2025, 10.1007/s40295-025-00506-6.

[38] M. Müller, "Dynamic Time Warping," Information Retrieval for Music and Motion, pp. 69–84, Springer, Berlin, Heidelberg, 2007, 10.1007/978-3-540-74048-3_4.

[39] L. Perko, Differential Equations and Dynamical Systems, Vol. 7 of Graduate Texts in Mathematics. New York, NY: Springer-Verlag, 2nd ed., 1996.