A VISUAL ANALYTICS APPROACH
TO PRELIMINARY TRAJECTORY DESIGN

Wayne R. Schlei; Kathleen C. Howell†

Recent developments in astrodynamics suggest a wealth of design potential within the context of the circular restricted three-body problem. Exploitation of the expanding dynamical and mathematical insights, though, has been difficult to capture within a real-time design setting. Emerging from the ability to represent large amounts of information through visual environments, visual analytics is a new science that focuses on the application of graphical depictions to facilitate discovery. Moreover, visual analytics blends the science of analytical reasoning with the implementation of interactive visual interfaces. In considering the most effective approach to incorporate visual elements in a largely automated process, this investigation blends the fundamentals of trajectory design in multi-body regimes with the implementation of visual analytics, thereby merging visualization tools, differential corrections algorithms, and the intuition of a knowledgeable designer into one expansive design approach. Visual analytics offers a basis for rapid investigation and design with access to a wider range of options for the construction of trajectories that meet mission requirements.

INTRODUCTION

Computer-generated visualizations offer an exceptionally powerful tool for the understanding of mathematical concepts and spatial relationships. In spacecraft trajectory design, visualization is frequently a key to success. In recent years, commercial software packages have emerged, such as Satellite Tool-Kit®, that support trajectory development with adaptable three-dimensional (3D) views and spacecraft flight animations from the perspective of a variety of coordinate frames. Recent advances in trajectory analysis for vehicles under the influence of multiple gravitational fields offer innovative concepts for scientific exploration, especially within the context of the Circular Restricted Three-Body Problem (CR3BP). Exploiting the dynamical structures available in multi-body regimes, such as the CR3BP, is very appealing for design scenarios, and, in fact, much effort is currently focused on developing numerical capabilities to enable faster adaptive design strategies. But effectively exploiting such structures within an interactive visual design environment remains an intriguing challenge. The best balance between automated computation and visual insights can be elusive.

In recent years, the emergence of visual analytics, a science that merges the intuition of humans with scientific visualization and interactive environments, assists researchers from various fields in solving multidimensional problems and examining complex systems. Astronomers and astrophysicists, for example, incorporate visual analytics while producing three-dimensional (3D) simulations of a supernova core collapse to analyze patterns in the plethora of output parameters from such an event including turbulence, rotation, radiation, magnetic fields, and gravitational forces. In fact, visual analytics is often employed to extract information and insight from large quantities of unstructured data, but visual analytics can also be incorporated in support of design decisions in the engineering realm. With visual analytics applied in the automotive industry, engineers interactively alter vehicle construction schemes and various parameters during 3D flow simulations to optimize air resistance or engine efficiency. A design approach incorporating visual analytics is analogous to an interactive computer-aided design (CAD) application for drafting mechanical parts. For

†Hsu Lo Professor of Aeronautical and Astronautical Engineering, Purdue University, School of Aeronautics and Astronautics, 701 West Stadium Avenue, West Lafayette, Indiana 47907-2045.
example, a mechanical designer uses the visual interface in a CAD package to sketch geometries, generate 3D representations of parts, and test-fit a product assembly. The visual interactive CAD tool offers a mechanical designer the ability to quickly assess the parts and the mechanical functionality based on visual feedback and, then, to interactively modify components to meet a variety of requirements.\(^3\)

In seeking a more automated trajectory design process, the most effective strategy to incorporate the visual elements is explored. Visual analytics may offer a framework. For example, the use of visualization simply in post-processing offers only visual comprehension. However, the rendering of a scene requires that data representing graphics primitives (triangles, polygons, objects) is accessible for culling or lighting operations.\(^3\)

A visual analytics design application can utilize this same information with human interaction; the visual interface can be exploited to construct and manipulate initial guess arcs for potentially seeding a differential corrections process thereby assisting efficiency in the trajectory design process. Furthermore, if the corrections process is implemented within the same graphical interface, the preliminary trajectory construction and analysis occurs instantly. In short, visual analytics combines interactive computation and a knowledge base with trajectory visualization and other scientific visual techniques, creating an interactive approach for generating solutions and allowing other design options.

**Previous Contributions in Trajectory Design**

One commercial package that is available for spacecraft trajectory design in libration point missions is Satellite Tool-Kit\(^R\) (or STK\(^R\) made by Analytical Graphics, Inc.), which includes a visual component. Both STK\(^R\) and its predecessor Swingby have been successfully employed during the design phase and for flight operations in support of various libration point missions including Wind, SOHO, ACE, MAP, and CON-TOUR.\(^1\) The STK\(^R\) developers and users continually emphasize the advantages of design analysis with immediate visual feedback. The communication of trajectory concepts is greatly enhanced with 3D imagery and animations.\(^1\) Design within the context of the CR3BP also offers an opportunity to apply dynamical systems and orbit stability analysis tools. For example, the role of invariant manifolds and their applications to trajectory design are introduced by various investigators including Lo, Anderson, et al.,\(^4\)–\(^6\) and the effective use of Poincaré maps for trajectory design applications is demonstrated by numerous researchers.\(^6\)–\(^8\)

Some previous investigations introduce elements of visual analytics into the analysis of design problems. For instance, to transfer from a specified low-Earth orbit to a Sun-Earth L\(_1\) halo orbit, Museth, Barr, and Lo use a 3D workbench with an immersive interface to select a suitable arc along a stable manifold based on visual inspection.\(^9\) The resulting transfer arc is then employed as part of the design for the ‘Terrestrial Planet Finder’ mission.\(^9\) The implementation by Museth, Barr, and Lo introduces the visual design concept; however, more extensive functionality is directly available through a visual design approach. The purpose of the application of visual analytics to the spacecraft trajectory design problem is the potential to introduce the most recent theoretical developments into the design process as quickly and effectively as possible. In the current effort, the goal is the exploitation of advanced dynamical systems theory to generate solutions that meet specific trajectory design objectives that may be difficult to quickly achieve otherwise. As a designer or analyst, the goal is the most effective use of this new wealth of visual information while simultaneously shifting towards a more automated design approach.

**Current Work**

The design and construction of spacecraft trajectories in multi-body environments may be enhanced with the application of a visual analytics framework. The use of visualization is more than a simple display of the trajectory information; it also supplies trajectory development tools and new information concerning the dynamical models. Implementing user-built computation modules and interaction functions with visual interfaces allows the user to analyze the available information and immediately adjust the data, visualization, or computation processes. The visual elements can, in fact, support the automated processes. As a potential strategy, this investigation offers a blend of spacecraft trajectory design with visual analytics, thereby merging visualization tools, numerical algorithms, and designer intuition into one design approach.

---

\(^2\) Museth, Barr, and Lo (2005)
\(^3\) Visual analytics (2018)
\(^6\) Museth, Barr, and Lo (2002)
\(^7\) Museth, Barr, and Lo (2003)
\(^8\) Museth, Barr, and Lo (2004)
\(^9\) Museth, Barr, and Lo (2005)
VISUAL ANALYTICS

With the advent of advanced computer graphics and rapid rendering software and technologies, the ability to represent large amounts of information through visual environments has evolved. Thus, a new science born of this ability focuses on the effective application of graphical depictions to facilitate discovery. Visual analytics blends the science of analytical reasoning with the implementation of interactive visual interfaces. Keim et al. define visual analytics more precisely as “an interactive process that involves information gathering, data preprocessing, knowledge representation, interaction and decision making”. The computing and visualization power of machines combined with the decision-making abilities of humans leads to knowledge discovery that may not be available through standard analytical tools. The application of visual analytics to spacecraft trajectory design supplies an immensely powerful tool in research and development as well as enhanced capabilities to produce a wider range of options for trajectories to meet mission requirements. More effective use of the visual components can significantly aid a trajectory analyst.

The Visual Analytics Process

Understanding the application of visual analytics to trajectory design begins with a detailed description of the visual analytics process. The goal of this process is an innovation or insight, $I$, based on some initial input data, $S$. Three phases of analysis exist in transforming $S$ to $I$, including analytical abstractions $A$, hypotheses $H$, and visualizations $V$. All the component functions and their interactions are demonstrated in Figure 1, which is a diagram of the visual analytics process. The analytical abstraction phase, identified by the gray rectangle in Figure 1, involves tailoring the input data into manageable or relevant blocks through data manipulation techniques and the applications of theory. Characterized by the purple rectangle in Figure 1, the hypothesis phase reflects a type of confirmation analysis in the form of a supposition that is evaluated, similar to a hypothesis in the standard scientific method. However, in the visual analytics process, the affirmation of new ideas is drawn from visual components along with previously existing theory and data. Thus, the core of the visual analytics process is the implementation of the visualization phase. This visualization phase, represented by a blue box in Figure 1, incorporates elements of scientific visualization, information visualization, and the cognitive and perceptual sciences as well as graphical interfaces for visual exploration of information. Overall, the visual analytics process is more encompassing than simply the application of visualization methods. It is actually a coalescence between visualization, human cognition and interaction, as well as theoretical and data analysis.

Figure 1. Diagram of the visual analytics process (adapted from 2, 11, 12).
Fundamentally, visual analytics transforms an input $S$ to an insight $I$. Thus, the visual analytics process is formally represented by the mapping function $F : S \mapsto I$. The over-arching transformation $F$ consists of a set of sub-functions $f$, where $f \in \{A_W, V_X, H_Y, U_Z\}$. This set maps the visual analytics process from one phase to another as illustrated by Figure 1. The subscripts $W, X, Y, Z$ are used as shorthand notation to symbolize different types of functions utilized in each member of the set $f$. Thus, each member of $f$ is a further subset of functions that are described by the following:

- **Functions $A_W$**: Data preprocessing and theoretical analysis tools reside in the set of functions $A_W$ that map the input data to an analytical abstraction (i.e., $A_W : S \mapsto A$). The subscript $W$ symbolizes each type of function available in the mapping from $S$ to $A$, such that $W \in \{T, C, L, Th\}$. Transformations of data are contained in $A_T$, data cleaning operations reside in $A_C$, $A_L$ represents functions used to select specific sub-sections of the input data, and the applications of analytical theory are included in $A_{T_H}$.

- **Functions $V_X$**: The set $V_X$, where $X \in \{S, A, H\}$, denotes the employable visualization functions. Visualizing the data directly is performed with the mapping $V_S : S \mapsto V$, an analytical abstraction is viewed via $V_A : A \mapsto V$, and $V_H : H \mapsto V$ represents the transfer function denoting the visualization of a hypothesis. These visualization functions cover rudimentary computer graphics procedures (e.g., representing a point as a sphere or viewing axes and flow meshes as grids) and embrace any known scientific or information visualization algorithms.

- **Functions $H_Y$**: The formation of a hypothesis occurs within the functions $H_Y$, where $Y \in \{A, V\}$. Hypotheses generated from analytical abstractions of the input data $S$ are symbolized by $H_A : A \mapsto H$. Then, $H_V$ characterizes a hypothesis derived directly from a visualization ($H_V : V \mapsto H$).

- **Functions $U_Z$**: User interaction in the visual analytics process is represented with the functions $U_Z$, each sub-function identified by $Z \in \{A, H, V, CV, CH\}$. These interactive functions consist of either direct interaction with an object existing within a visual environment or as a decision-making component. The set of functions $U_A : A \mapsto A$ includes all selections or alterations a user implements with an analytical abstraction. A hypothesis can be modified by the user through the function $U_H : H \mapsto H$. Adjustments to a visualization via human interaction are represented by functions linked with $U_V : V \mapsto V$. Basic camera operations like zooming, panning, or view reorientation, alterations to graphical object properties such as color and transparency, as well as lighting styles are all associated with $U_V$. The desired output of the entire process, i.e., the insight $I$, is then deduced as the output from either the visualization ($U_{CV} : V \mapsto I$) or hypothesis ($U_{CH} : H \mapsto I$) phases.

The steps from $S$ to $I$ are then modeled in terms of these components.

The entire visual analytics process is recursive, based on the user-specified level of detail. The output of this process, or the realization of the insight $I$, may sometimes suggest that either further knowledge is available or more detail is required to be sufficient to render a conclusion. If so, modifications of an individual phase or a different set of transfer functions are derived given the current level of insight. In such a scenario, the process restarts with a feedback loop, $G$, that maps the achieved insight $I$ back to new input data $S$ ($G : I \mapsto S$). The feedback function $G$ also appears in Figure 1. This feedback loop reflects an update to the input, that is, a modification of the input data $S$ resulting from a discovery. (For more information on the visual analytics process and applications, see Keim et al., Risch et al., or Huang and Nguyen.)

**Selection of an Interactive Visualization Software**

Visualization software options are expanding rapidly. In addition, computational software that offers visual support is widely available including packages for astrodynamics applications. For this investigation, more extensive access to interactive visualization capabilities is necessary since user interactions are an integral part of the visual analytics process. Interactive visual explorations require a visual environment capable of fast navigational changes, such as zooming, panning, and reorientation of the camera, to grasp spatial location and the depth of objects in a three-dimensional (3D) scene. Also, real time human-computer
interaction involves visualization algorithms capable of processing hundreds of thousands of visual modifications in seconds to be useful for knowledge discovery (with a maximum of only a few minutes). If a slight modification is desired, for instance, a change in coloring, the individual objects must be modified quickly – inside the visual environment – for real-time knowledge discovery. Thus, the visualization software must possess the dexterity to implement quick modifications of a scene. Any sort of data computation involved with the analytical abstraction phase is also subject to the same speed requirements for real-time insight. It is likely that user-built functions are responsible for abstracting or analyzing a specific substructure of the data, but the visualization tool may also incorporate some computational components that support this concept. The scientific visualization suite Avizo© from Visualization Sciences Group (VSG) is one particular visualization package that offers the versatility to supply all of the required elements of the visual analytics process. Stalling et al. state that Avizo© is built with flexibility, interactivity, and extensibility as design goals. Avizo© incorporates a variety of different data types with the ability to visualize and perform operations with multiple data sets simultaneously. It also houses an enriched interactive capability with 2D and 3D environments. Human interaction with a visual scene is accomplished by using picking functions and dragger objects that are built using Open Inventor©. Information about points, a mesh grid, or the faces of a surface, as well as any associated data values, are obtained by simply clicking. Thus, Avizo© serves as the basis for the visualizations in this analysis.

The Visual Analytics Process Applied to Trajectory Design

The visual analytics process is adapted from the general process in Figure 1 to accommodate the various elements of a sample trajectory design scenario. First, the mission parameters and requirements supersede the input data, S, as the starting point in the visual analytics process. Examples of possible trajectory parameters include predetermined starting or ending conditions, timing conditions, and known mission objectives either operational or scientific in origin such as rendezvous, landing, desired ground tracks, or predetermined orbits. Figure 2 indicates mission parameters by the red component that initiates the visual analytics process. For the analytical abstraction phase, A, the trajectory design supplement represents the subdivision of a mission into pertinent phases, the selection of a dynamical model for propagation, the application of known dynamical theory, as well as the formulation and execution of corrections or computational procedures. The visualization phase, V, contains all the available graphical possibilities for points, vectors, and paths as well as any other scientific visualization procedures applicable to trajectory design. During the hypothesis phase, H, possible orbits or transfer arcs that may meet mission criteria are proposed. Any adjustments to the corrections process are also theorized during the hypothesis phase. The resulting insight, I, is then a partial path or the complete spacecraft trajectory. As in the original form of the visual analytics process, I might also include discovered knowledge such as some particular behavior associated with a dynamical model. It is possible that a resulting trajectory, I, may not meet the mission requirements set forth by S. In such a case, a feedback loop, G, adjusts A, V, or H to determine a viable path. Finally, G can also modify S if the constraints on the design are infeasible.

The functions mapping one visual analytics phase to another phase now apply to direct trajectory design procedures. The interplay between the visual analytics mapping functions and trajectory design appears in each of the function types:

- **Functions A_W**: The set of functions A_W map mission and/or trajectory parameters into useful analytical abstractions. Note that S may also contain any discovered trajectory or insight (I) from a previous application of the visual analytics process. Each particular subfunction in the set A_W represents relevant trajectory design procedures. The A_T functions modify states using spatial transformations such as translation, rotation, and scale, or the functions can extract the velocity states of a trajectory for examination in a phase space. Moreover, A_T functions map trajectory constraints into the differential corrections process; functions in this set also alter selected states in position, velocity, and time through the execution of a differential corrections algorithm. The removal of states, arcs, constraints, and mission objectives are types of data cleaning operations, A_C. The subset of functions in A_L are associated with the selection of subsets of data; functions in the set A_L can separate a mission into phases, isolate a particular phase for analysis, select a specific arc from a group or family, or abstract patch points
Figure 2. The visual analytics process adapted for a mission design scenario.

for differential corrections. Applying analytical theory in the design process then constitutes the set of functions in $A_{Th}$.

- **Functions $V_X$:** Visualization mapping functions display relevant information about the various steps in the visual analytics process to the user. The visual representation of $S$ ($V_S$) features specific trajectory information that is initially introduced. For example, $V_S$ may include the visualization of specific states, rendezvous orbits and conditions, communications constraints such as coverage cones, desired B-plane zones for flybys, error or landing ellipses, as well as timing conditions that can appear via animations. Analytical abstractions, viewed with the functions in the set $V_A$, are typically realized as point clouds and vectors to represent data at selected states or in specified pointing directions. A hypothesis depiction ($V_H$) assists a trajectory designer to immediately assess the feasibility of, for example, a transfer concept, a new constraint, or a correction adjustment. Also, each of the $V_X$ functions support coloring options and potential animations for enhanced clarity.

- **Functions $H_Y$:** From the analytical abstractions in Figure 2, an experienced trajectory designer employs the various $H_A$ functions to hypothesize concerning the orbit or transfer type that will most likely support the mission characteristics identified in $S$. Thus, one example of an $H_A$ mapping is the selection of a set of state elements that represents a transfer arc. A trajectory designer may also formulate an adjustment to a corrections procedure by observing the intermediate arcs before convergence or a textual readout of the iteration history. These theorized adjustments from visual feedback are representations of a hypothesis deduced from a visualization, or $H_V$. Graphical insight may also trigger new transfer ideas.

- **Functions $U_Z$:** User interaction binds trajectory design theory and methodologies to the visual analytics process. In fact, if the subset of user interaction functions $U_A$, $U_V$, and $U_H$ are incorporated into a single visualization package, a trajectory designer could potentially create a preliminary end-to-end trajectory concept within a real-time design session. With the available $U_A$ functions, a user can interactively apply astrodynamical theory, generate solution arcs for various trajectory or mission phases, as well as select and manipulate trajectory data to form an initial guess. In other words, the trajectory can be spliced together from various models and theoretical formulations to form a starting point for some corrections process. The designer could subsequently structure a differential corrections algorithm inside the visual environment, allowing the user to include even spontaneous constraints and adjustments. Interactive adjustments to the visualization ($U_V$) capture the spatial relationships between arcs and highlight trajectory information relevant to constraints. Rapid changes to any transfer concept
are inserted immediately using the attached computational algorithms. To complete the path from $S$ to $I$, the $U_{CV}$ and $U_{CH}$ functions represent any user conclusions about the viability of the generated trajectory and the constraints as well as decisions or decision trees for proceeding if the current result is unsatisfactory.

The application of visual analytics to this design process implies a design procedure encompassed in one operation that unites visualization tools, interactive components, real-time computations, and a decision-making ability.

With all of the associated functions, this process represents an interactive bridge between visualization and the trajectory design process. Current software products for trajectory development (e.g., STK® and FreeFlyer®) already supply some visual analytics components. Accompanied with impressive planetary models and detailed surface maps, these design packages offer visual support for point locations (for ground sites and satellite positions) and trajectory arcs referenced to various celestial coordinate frames with graphical markers and line segments. Also, these packages illustrate constraints through graphics primitives such as triangles, circles, cones, ellipsoids, etc. Human interactions are also incorporated into these software products. For example, user-controlled view navigation, frame and model selection, as well as trajectory propagation are available.

More experienced trajectory analysts may benefit substantially from more direct human interaction with the visual environment, particularly in research-oriented projects. The ability to adjust visual representations of objects, such as mapping color, transparency, or the size of graphics objects to germane quantities, reveals more information about a design problem. Also, design with visual analytics uses the visual objects and tools to automatically “grab” the numerical data representing desired choices and implements the results instantly. Thus, the appeal of visual analytics to advanced users is then the real-time implementation of designer experience during analysis with the assistance of comprehensive visualizations.

**TRAJECTORY DESIGN TOOLBOX**

Although the visual analytics process can be employed in any astrodynamical system, this investigation demonstrates the concepts within the context of the CR3BP. The motion of interest in the CR3BP focuses on the path of a spacecraft under the influence of two gravitational masses (or primaries). The spacecraft does not influence the motion of the primaries, and it is assumed that the primaries orbit the barycenter, $B$, in a circular fashion. A rotating reference frame, $R(\hat{x}\hat{y}\hat{z})$, is convenient for modeling the motion in the CR3BP with respect to $B$. The $\hat{x}$ axis resides on the line adjoining the primaries and points from $B$ to the smaller primary. The positive $\hat{z}$ axis is parallel to the angular momentum vector of the primaries about $B$, and the $\hat{y}$ axis completes the right-handed triad. The CR3BP possesses five equilibrium solutions, or libration points, with three collinear points ($L_1, L_2, L_3$) and two equilibrium points ($L_4, L_5$). Also, one integral of motion exists—the Jacobi constant ($C$). The CR3BP model is incorporated by numerous researchers for design of periodic orbits about libration points and transfer path designs of various types.8, 14, 15

Trajectory solutions in the multi-body gravitational environment are achieved inside a visual scene by employing differential corrections schemes and methods from dynamical systems theory as computation modules. The differential corrections module implements multiple shooting schemes with options to either hold the integration time between segments constant (a fixed-time method) or to incorporate the integration time as a design variable (a variable-time method). Constraints can be placed on a solution that fix the starting position or the final position along a trajectory.16 Another targeting algorithm with a periodicity constraint allows users to compute a wide variety of periodic orbits.16 Users can also implement stable and unstable manifolds for unstable periodic orbits. Even Poincaré maps are available as a tool for generating solutions inside a visual environment.

The application of visual analytics to trajectory design requires the interactive manipulation of trajectory data. The initial guess in differential corrections algorithms is generated by piecing together arcs and states. For a visual analytics approach, interactive tools supply a platform to visually construct trajectory arcs and create an initial guess for differential corrections procedures. In a visual environment, a designer performs
trajectory manipulation through a graphical user interface (GUI). The manipulation of trajectory states is housed in several groups of functions that are members of the $U_A$ set.

**Combinatorial Operations** One set of interactive tools that are very useful for displaying significant trajectory information and assembling or modifying an initial guess are combinatorial operations. These operations consist of algorithms that link trajectory segments together or decompose a trajectory arc into segments. For manual input without the aid of a visual environment, this step is usually accomplished with indexing arrays. For example, the first 20 states corresponding to a 100-state trajectory append to another 50 states from a different trajectory arc, resulting in an array of 70 trajectory states that are ready for a differential corrections process. In a visual environment, however, trajectory segments are assembled with user interaction, visual cues, and the automatic combination of trajectory objects.

**“Stacking” Periodic Orbits** Another particularly useful operation for trajectory design is the “stacking” of periodic orbits. Multiple revolutions of a single orbit allocate time in trajectory planning for phasing concerns in a rendezvous or close-encounter problem or for observations to meet scientific objectives. Thus, the ability to repeat a specified number of orbits is key capability. The repetition of revolutions in a periodic orbit is denoted here as “stacking” since copies of the same orbit are overlaid in terms of position and velocity states but spread out in time. In other words, the “stacking” functionality mimics a spacecraft completing several revolutions of the same orbit.

**Point-Sampling** In a visual design scenario, trajectories are numerically integrated and designed in real-time, so the construction of a set of patch points comprising an initial guess for targeting applications occurs concurrently. Typically, numerical integration techniques incorporated in many astrodynamics applications produce a sufficient number of points for a smooth visual representation by rendering a set of linearly connected graphics primitives (e.g., line segments, rectangular prisms, or cylinders). However, the number of points generated in a numerical simulation can impede progress when employed as patch points in a numerical corrections algorithm. In a multiple-shooting algorithm, the memory requirements and the number of operations required for a solution increase as the number of patch points increase. Sparse matrix algorithms can reduce memory and cost requirements, but the simplest approach is to minimize the number of patch points. Sampling is based on visual observations including curvature and regions likely to be numerically sensitive. A point-sampling operation may equally distribute points based on some parameter, typically time, or sample directly from an integrated result. The implementation of a sampling algorithm allows the user to construct and modify patch points, supplying an interactive functionality with targeting procedures.

**Arc Selection** Some numerical techniques that are incorporated into trajectory design algorithms compute a collection of trajectories, such as continuation schemes for orbit families and manifolds emanating from various fixed points along a periodic orbit. Thus, the collection of trajectories resulting from the application of these numerical techniques presents a designer with a multitude of options for proceeding forward with a tentative trajectory concept. Although techniques that generate a collection of trajectories offer many options, most applications in trajectory design require only one path to proceed. A selection algorithm allows a user to employ a pick-callback (i.e., a mouse click) to extract a desired trajectory arc from a group, thereby supplementing the available $U_A$ function set. Museth, Barr, and Lo implement this interactive selection concept for designing manifold transfer paths to periodic orbits, but other types of investigations also benefit from this technique. This selection functionality also permits the re-use of previously computed data. If a large family of orbits is available for a specific system, a trajectory designer can incorporate any orbit from this family by clicking an orbit of choice. Therefore, manifold structures, orbit families, and any previously computed trajectory may be re-used in a future design or investigation.

**Spatial Transformation** One particularly useful aspect of an interactive visualization software package, such as VSG’s Avizo®, is the built-in ability to interactively transform data objects. Stalling et al. state that “visual environments should allow the user to easily transform individual datasets [or data objects] spatially with respect to others”. Thus, the visual software environment employed (i.e., Avizo®) offers a user interface to interactively modify the spatial location, orientation, and scale of objects. Interactive affine transformations are applied in a 3D scene using Open Inventor®; the transformations allow the dragging of objects that supply visual indicators for absolute or relative translations, rotations, and scaling operations.
The interactive transformation of trajectory arcs and patch points must transform the position and velocity states throughout the entire path. After a spatial transformation, the transformed states likely form a discontinuous solution within the context of the gravitational model. However, the initial guess is not required to be continuous since differential corrections processes enforce continuity.

**MULTI-BODY TRAJECTORY DESIGN SCENARIO**

When applied to trajectory design scenarios, a visually-interactive design philosophy offers the analysis of particular mission phases or entire end-to-end trajectories with complete maneuver itineraries. The benefits of design completely within a visual environment are particularly notable in multi-body models where no analytical solution is available. Real-time design of preliminary trajectories in the CR3BP, for example, within a visual environment allows an alternative design experience.

A visual trajectory design strategy is demonstrated through preliminary analysis in the Earth-Moon (E-M) CR3BP. Consider a sample design objective to generate a halo orbit in the vicinity of the libration point \( L_1 \). The creation of a halo orbit is initiated by loading a set of numerically-constructed \( L_1 \) Lyapunov orbits. An orbit close to the bifurcation between the Lyapunov and halo families is selected with a mouse click. The selected orbit is then extracted from the set and highlighted to register the selection (Figure 3(a)). For a smooth visual representation, a large number of points initially represents the selected orbit. This representation of the selected orbit is instantly sampled to a reasonably small number of points, equally spaced time, that generate patch points for a differential corrections process. The sampled states from the Lyapunov orbit are illustrated in Figure 3(b) as red spheres, representing the state vector positions, and red vectors, indicating the velocity directions.

![Image](image.png)

**Figure 3. User interaction during a preliminary trajectory design scenario in the E-M CR3BP involving the selection and point sampling of a periodic orbit.**

Since the extracted states are available to generate the visual scene, a spatial transformation is used to construct the initial guess for computing a periodic halo orbit. An interactive spatial transformation of the states is achieved by using additional graphical elements that a user can manipulate inside the visual environment. Thus, an initial guess for a halo orbit, as it appears in Figure 4(a), is produced by rotating the states corresponding to the Lyapunov orbit counterclockwise about the mean \( \hat{y} \) axis. The dashed line represents the trace of the original orbit after an interactive rotation in the scene, and the purple points and vectors reflect the transformed states. Clearly, the new states no longer produce a continuous trajectory when numerically propagated as demonstrated by the purple integrated arcs in Figure 4(b). However, the transformed states serve as a sufficient initial guess to generate the halo orbit. Thus, a user-created module for computing periodic orbits is applied to the transformed states. This computation tool for periodic orbits utilizes an asymmetric, variable-time multiple-shooting scheme. The resulting structure, appearing as the green orbit in Figure 5(a), is a periodic halo orbit near the E-M \( L_1 \) point. Quasi-periodic behavior exists in the close vicinity of
this halo orbit,$^{15}$ thus, a quasi-periodic structure is also computed quickly using interactive manipulation of the generated data in conjunction with corrections procedures. Eight revolutions of the halo orbit are quickly “stacked” (with four patch points per revolution), and the initial point is translated in the $-\hat{z}$ direction to construct the green initial guess in Figure 5(b). Instant implementation of a targeting algorithm with a fixed initial position constraint is then applied to the green initial guess and converges on the blue, quasi-periodic structure in Figure 5(b).

The design scenario is next extended to plan a transfer trajectory arc from an initial Earth orbit to a quasi-periodic, near-halo trajectory in the vicinity of $L_1$. Assume that a spacecraft is launched into an initial orbit, appearing as the red trajectory in Figure 6, with the orbital elements $a = 22,000$ km, $e = 0.3$, and $i = 28.5^\circ$. Twenty revolutions ($\approx 8.7$ days) are propagated from an initial state (red sphere in Figure 6) to examine departure locations for transfers to the $L_1$ vicinity. The departure position is selected arbitrarily by sliding the gray sphere representing the departure location along the orbit. This departure point indicates that a maneuver is implemented at this location to initiate a transfer arc, but the maneuver and the transfer arc are currently unknown.
Figure 6. The initial Earth orbit for a mission to a quasi-periodic trajectory in the vicinity of \( L_1 \) in the E-M system.

The transfer design process originates with a preliminary concept. An initial transfer option is constructed by incorporating orbital stability analysis, manifold propagation, and a maneuver testing tool. Stability information is obtained from the same periodic orbit generation tool that is employed to compute the halo orbit in Figure 5(a); the halo orbit in Figure 5(a) is unstable and possesses a stable and unstable subspace. The instability suggests that an asymptotic approach to the halo orbit, and the nearby quasi-periodic structure, exists. Therefore, the stable manifold approaching the halo orbit is determined using a computation module that is tailored with options for manifold generation such as the type of manifold (stable or unstable), the propagation time, and the number of fixed points along the path. For this halo orbit, the computation module propagates the stable manifold for \( \approx 34.74 \) days employing 50 fixed points along the orbit. Subsequently, the stable manifold is added to the visualization (displayed in Figure 7). Unfortunately, there is no intersection between the stable manifold that approaches the halo orbit and the initial orbit, indicating that an intermediate transfer arc is required to bridge the gap.

Figure 7. Projection of the stable manifold (cyan) associated with a halo orbit (green) propagated in reverse time for 34.74 days.

The determination of an intermediate transfer is then achieved by testing potential maneuver vectors at an arbitrarily selected departure location on the initial orbit. At the gray departure location, some sample maneuver options are examined and displayed in Figure 8. A possible intermediate arc (black trajectory in Figure 8(b)) intersects many manifold trajectories associated with the halo orbit. A manifold insertion location is then visually selected where the intermediate transfer and the stable manifold trajectories intersect. Since several intersection locations are available, the intersection with the smallest change in the velocity direction between the intermediate transfer arc and the stable manifold trajectory is selected by visual inspection. The intersection point (the purple sphere in Figure 9) is representative of a manifold insertion location and is, therefore, employed to isolate the portion of the intermediate transfer arc leading to stable manifold insertion
Figure 8. Potential intermediate transfer arcs (black) explored through testing various maneuvers at the departure location (gray) on the initial orbit (red).

(roughly 3.5 revolutions of the transfer orbit). The final phase of the transfer is selected from the representative collection of stable manifold trajectory arcs in Figure 9. The unused segment of the selected stable manifold arc is snipped at the manifold insertion point. With the truncated intermediate and stable manifold arcs, an initial guess for the transfer trajectory is accomplished.

Figure 9. Views of an interactively constructed initial guess for a trajectory from an Earth orbit (red) to a quasi-periodic orbit (blue) in the vicinity of \( L_1 \) in the E-M system.

An entire end-to-end trajectory is generated by implementing multiple shooting schemes with the interactively generated transfer guess. The proposed initial guess includes a departure maneuver from the initial orbit and a manifold insertion maneuver. The design is split into two shooting problems with a matching point at the manifold trajectory insertion location: one problem computes adjustments to the intermediate transfer arc (Transfer Leg 1), and the alternate problem addresses the asymptotic approach to a quasi-periodic orbit in the vicinity of \( L_1 \) (Transfer Leg 2).

Since the manifold insertion point is not precisely on the stable manifold associated with the final quasi-periodic orbit, Transfer Leg 2 is computed first. The stable manifold arc is sampled to produce ten patch points using a direct sampling from the numerical propagation. Subsequently, the 32 patch points originally generated to compute the quasi-periodic orbit in Figure 5(b) are appended to the first ten patch points from the stable manifold arc to create an initial guess for Transfer Leg 2 (represented in Figure 10(a)). A multiple shooting computation module determines a continuous solution for the second transfer leg by employing a variable-time approach to enforce only continuity constraints. The resulting converged transfer, appearing as the gold arc in Figure 10(a), flows freely into a quasi-periodic solution in the vicinity of \( L_1 \). The resulting Transfer Leg 2 solution varies less in size than the initial guess (blue), but still achieves the objective. As
a result of the corrections procedure, the manifold insertion location shifts slightly to permit an asymptotic approach.

The first transfer leg is then constructed following the determination of the asymptotic approach into the target orbit structure. The starting position that defines the converged Transfer Leg 2 (i.e., the manifold insertion location) is incorporated as a final position constraint for computing Transfer Leg 1. The initial guess for the first transfer leg (black) in Figure 9 is sampled and approximated as 29 states, including the departure location. The starting state corresponding to Transfer Leg 2 is then appended to these states to form 30 patch points for the differential corrections process. Incorporation of a variable-time, multiple shooting algorithm supplies the converged solution for the first transfer leg (Figure 10(b)) if the initial position is fixed at the departure maneuver and the final position is constrained to the manifold insertion location. As evident in Figure 10(b), the converged Transfer Leg 1 possesses the same general shape as the initial guess with three phasing loops before the manifold insertion maneuver.

After the construction of both transfer legs, the visual analytics design approach renders the entire path from the initial orbit to the $L_1$ quasi-periodic orbit. Obviously, this design approach creates a visual representation of the complete trajectory inside a visual environment where the viewing angle is easily adjusted (Figure 10(c)). Also, the timing and maneuver details (summarized in Table 1) are obtained since that information is available for computing continuous solutions with multiple shooting algorithms. The total transfer from the departure maneuver to arrival at the quasi-periodic orbit is $\approx 19$ days with a total $\Delta V$ equaling 2.81765 km/s. This design is not necessarily fuel- or time-optimal, but it is constructed in one operation.

Figure 10. Visual analytics design scenario for a transfer trajectory from an Earth orbit (red) to the quasi-periodic orbit (gold) in the vicinity of $L_1$ in the E-M system.
using the visual analytics process for trajectory design. Additional constraints, such as timing or maneuver magnitude bounds, can be easily incorporated. Since all the numerical data is available, subsequent analysis is facilitated. (Alternative multi-body design scenarios employing a visually-interactive process are available in Reference 16.)

Table 1. Trajectory details for a transfer trajectory in Figure 10

| Trajectory Event          | \( t_{\text{Start}} \) (days) | \( t_{\text{End}} \) (days) | \( \Delta t \) (days) | \( ||\Delta V|| \) (km/s) |
|--------------------------|-------------------------------|-------------------------------|------------------------|--------------------------|
| Starting Point           | 0                             | 0                             | —                      | —                        |
| Initial Orbit            | 0                             | 8.685                         | 8.685                  | —                        |
| Departure Maneuver (\( \Delta V_1 \)) | 8.685                         | 8.685                         | —                      | 1.21834                  |
| Transfer Leg 1           | 8.685                         | 16.718                        | 8.033                  | —                        |
| Manifold Insertion (\( \Delta V_2 \)) | 16.718                        | 16.718                        | —                      | 1.59931                  |
| Transfer Leg 2           | 16.718                        | 27.574                        | 10.856                 | —                        |
| Arrival                  | 27.574                        | 27.574                        | —                      | —                        |
| Quasi-Periodic Orbit     | 27.574                        | 134.542                       | 106.970                | —                        |
| Total                    | 0                             | 134.542                       | 134.542                | 2.81765                  |

INTERACTIVE IMPLEMENTATION OF POINCARÉ SECTIONS

Poincaré sections are a powerful tool for trajectory design within the context of multi-body models, but the computational cost is generally quite high due to the large volume of numerical integrations. In fact, interactive generation of a Poincaré map in a series or even in a multi-core CPU parallel computational framework is not always practical in real-time trajectory design. However, recent access to the parallel computing power of graphics processing units (GPUs) has created a new avenue for map generation. The construction of Poincaré sections through general-purpose computation on GPUs and the corresponding interactive elements offer more immediate insight. Interactive capabilities permit a transient definition of any Poincaré section that can potentially unlock new options in multi-body regimes.

Interactively Defining a Poincaré Section

A surface of section can be defined in terms of a vast set of hyperplanes, but the scope in this investigation is limited to one particular type. A Poincaré map is defined by the returns of a trajectory to a specific hyperplane, \( \Sigma \). If a real space is defined in terms of dimension \( N, E \in \mathbb{R}^N \), any continuous hypersurface can define the mapping for a section as long as the dimension of the hypersurface is less than \( N \) and the hypersurface is transversal to the flow.\(^{21}\) In the full CR3BP model, the equations of motion represent a six-dimensional (6D) space, so \( \Sigma \) can be defined as any hypersurface in 5D space (e.g., 1D curve, 2D surface, or a 3D object). Here, the hyperplane is restricted to a 2D plane in configuration space to permit interactive definition in a visual scene.

The hyperplane \( \Sigma \) is related to the rotating coordinate frame in the CR3BP through a rotation and a translation. The frame \( R (\hat{x} - \hat{\gamma} - \hat{z}) \) is consistent with the rotating frame in the formulation of the CR3BP. A second coordinate frame, \( P \), is then defined to be fixed on \( \Sigma \) with the origin at point \( O \), and the direction cosine matrix \( R_{LP} \) describes the orientation of \( P \) relative to \( R \). A schematic in Figure 11 conveys the relationship between the \( R \) and \( P \) frames. The normal to the plane \( \Sigma \), i.e., \( \hat{n} \), identifies the first axis in the frame \( P \). If point \( O \) is coincident with the barycenter \( B \) and \( P \) is initially aligned with \( R \) (such that \( R_{LP} = I \)), the axis \( \hat{n} \) is parallel to the direction of \( \hat{x} \) and the second axis in \( P, \hat{p} \), aligns with \( \hat{y} \). The unit vector \( \hat{q} \) completes the right-handed triad in the \( P \) frame. Also, the frame \( P \) may be translated relative to the barycenter \( B \) to incorporate planes in the rotating frame that do not intersect \( B \). The vector \( T \) represents the translation of \( \Sigma \) from the barycenter, \( B \), to point \( O \). When the translation vector \( T \) and the direction cosine matrix \( R_{LP} = I \) are defined, then \( \Sigma \) is constructed by applying the corresponding rotation and translation relative to the \( R \) frame.
User interaction defines the hyperplane $\Sigma$ through a graphical transformation in a visual environment. With $\Sigma$ represented by a graphical primitive, the transformation editor for shifting spatial data objects is employed to allow the user to interactively transform the rectangular polygon object. Therefore, the translation vector, $T$, and the rotation matrix, $R_L^P$, are defined through user interaction. The corners of the rectangular polygon are also scaled with the transformation editor, shaping $\Sigma$ as either a rectangle or a square of an arbitrary size. These translation, rotation, and scaling operations are standard transformation elements in computer graphics for relating a local graphical object frame to a global coordinate frame. In this investigation, the graphical object is the rectangular polygon symbolizing $\Sigma$, as viewed in the rotating coordinate frame, defining $P$ as the local graphical object frame and $R$ as the global coordinate frame. The interactively-defined plane $\Sigma$ then exists as input for a computation module that constructs Poincaré sections.

The selection of initial conditions to seed a Poincaré map is also based on the interactive definition of the hyperplane. After a user interactively transforms the graphical object representing a hyperplane, a structured grid of position coordinates that reside on the planar graphical object is outlined by the user. Each position coordinate in the grid possesses a distinct velocity magnitude for a user-specified Jacobi constant value. An initial state is then obtained by designating a velocity state that is parallel to the plane normal ($\hat{n}$) with the distinct velocity magnitude. Grids of initial states are implemented in a regular fashion with equal spacing between grid nodes. (A detailed mathematically description is available in Reference 16.)

**GPU Computation for Poincaré Sections**

The capability for parallel computation that is available with graphics processing hardware (GPU) enables the implementation of Poincaré mapping techniques during a visual design process. Recall that visual analytics components require real-time computational speeds (i.e., up to a few seconds) for an interactive discovery process. Unfortunately, most Poincaré map calculations imply a struggle to meet real-time speeds due to the number and density of the numerical integrations that are necessary to observe the behavior of the local flow. The computation of a Poincaré section on a computer processor (CPU) is quite sluggish if the numerical simulation is implemented sequentially. However, parallel processing applications are ideal for generating Poincaré maps since each numerical integration involved in the generation of a section is independent. Fortunately, GPUs are designed for intensive, highly-parallel computations—a necessity for graphics rendering applications (e.g., color blending and shading operations). With recent interest in expanding parallel computing beyond the realm of gaming graphics, GPU manufacturers such as NVIDIA are releasing programming interfaces that can supply significant support for engineering problems. The parallel computational power of the GPU introduces a Poincaré map as a real-time, visual trajectory design methodology.

Numerical simulation and detection of $\Sigma$ plane crossings are achieved by simple algorithms for a smooth integration with CUDA™—a GPU programming language developed by NVIDIA. A Runge-Kutta 4th-order numerical integration scheme (RK4) is employed since the implementation involves straightforward, linear
computations that enhance the computational efficiency with CUDA™. The Poincaré map approximates returns to the plane $\Sigma$ via linear interpolation between numerical integration steps since the step size for an RK4 scheme must be fairly small ($h < 1 \times 10^{-5}$) to describe the motion in the CR3BP. For added versatility, the user may select to generate the map with either (i) single precision for greater computational speed, or (ii) double precision for enhanced accuracy.

A Sample Near-Earth Hyperplane in the Earth-Moon System

A simple Poincaré map is employed to illustrate the interactive capabilities with visual analysis in the CR3BP. Since the map generation techniques explore high-dimensional behavior by design, the hyperplane $\Sigma_1$ is selected to compute some planar initial conditions in conjunction with out-of-plane trajectories for comparison in the E-M system. Through user interaction, the plane $\Sigma_1$ is positioned in the $xz$ plane near the Earth on the $+\hat{x}$ side with the plane normal $\hat{n}$ directed opposite to $\hat{y}$, (i.e., $\Sigma_1 : -y = 0$). In Figure 12(a), a green square represents $\Sigma_1$ with one edge intersecting the $\hat{x}$ axis. An $8 \times 8$ grid (i.e., 64 initial states) seeds the Poincaré map and is also displayed in Figure 12(a) as colored arrows that are scaled by velocity magnitude. Again, the Jacobi constant value, pre-specified as $C = 3.12$ in this initial simulation, determines the velocity magnitude at each initial position. Each initial state on $\Sigma_1$ is assigned an unique color based on an index number and a rainbow colormap. The purple and dark blue initial states, therefore, represent planar initial conditions for the map, and all other initial condition vectors possess a positive $\hat{z}$ component in position. Also, preliminary intuition concerning the behavior is available through a rendering of the zero-velocity surface corresponding to this value of Jacobi constant. Viewed as a white surface in Figure 12(a), this $C$ value allows a wide range of motion including escape through the libration point gateways, implying a possible chaotic response.

Once the plane $\Sigma_1$ is placed appropriately through user interaction, a Poincaré map is computed with the GPU and examined within the visual environment. A user determines the number of returns and the simulation length. For this example, the two-sided Poincaré map for the $8 \times 8$ grid of initial states in Figure 12(a) is computed with a GPU simulation in under a few seconds, meeting the real-time computation speeds required by visual analytics. Up to 100 returns to the plane $\Sigma_1$, over a time span of 434 days, are captured with the GPU parallel computational processes; the results are subsequently visualized in configuration space (Figure 12(b)). A positive or negative crossing is easy to distinguish in Figure 12(b) since the positive returns ($\dot{y} < 0$) reside on the same side of the Earth as $\Sigma_1$. As is apparent in Figure 12(b), none of these initial states generate trajectories that leave Earth-vicinity, even though the $L_1$ and $L_2$ gateways are open. Visually

\begin{figure}[h]
  \centering
  \includegraphics[width=\textwidth]{figure12.png}
  \caption{The hyperplane $\Sigma_1$ and the associated Poincaré map for an $8 \times 8$ grid of initial states simulated for 434 days with up to 100 returns per initial state in E-M system ($C = 3.12$).}
\end{figure}
interesting trajectories are selected for analysis in configuration space using pick-callbacks within the scene. A user-selected point with a mouse click returns an index that is then back-traced to the corresponding initial state. The same initial condition is, in turn, simulated again but visualized as an entire trajectory over time. A planar \((xy)\) trajectory (purple curve in Figure 13(a)) demonstrates typical in-plane quasi-periodic behavior (i.e., the trajectory precesses about a primary in the rotating view but is not periodic). A second trajectory that originates with a small out-of-plane component is also selected for comparison (light blue trajectory in Figure 13(b)). The out-of-plane trajectory possesses quasi-periodic behavior that is quite similar to the in-plane quasi-periodic orbit, but the initial out-of-plane component tends to shape trajectories into a 3D “bowl” structure in configuration space as evident from two out-of-plane trajectories selected for observation in Figures 13(c) and 13(d). The generation of these trajectories from the Poincaré section representation allows a user to implement Poincaré maps as part of the visual analytics design process.

![Figure 13. Trajectories selected from the Poincaré section in Figure 12(b) \((C = 3.12\) in the E-M system).](image)

A Sample Hyperplane Near \(L_5\) in the Earth-Moon System

Another sample hyperplane in the Earth-Moon system is the basis for a map that resides in the vicinity of the \(L_5\) libration point. For this example, the hyperplane \(\Sigma_2\) is interactively translated to intersect \(L_5\) and oriented such that \(\hat{n} = \hat{x}\). Shaped as a rectangle in this example with the long dimension in the \(\hat{p}\) direction, the blue hyperplane in Figure 14(a) is defined as \(\Sigma_2 : x = 0.487512\). The Jacobi constant value is pre-specified as \(C = 2.96\), which pulls the zero-velocity surface completely out of the \(xy\) plane. Therefore, all the initial states that are defined on an \(8 \times 8\) grid on the plane \(\Sigma_2\), also apparent in Figure 14(a), exist in a valid region of flow. A quick GPU simulation generates a two-sided Poincaré map that evaluates up to 100 returns per initial state over 434 days. Visualizing the returns in configuration space as uniformly scaled spheres with
a unique color for each initial state (Figure 14(b)), the trajectories appear to traverse a vast range of space; returns are located far beyond $L_4$ and $L_5$ in the $\pm \hat{y}$ directions yet expand across a wide array of $z$ values.

The visualization is modified to display additional information from the map such that orbital structures in the vicinity of the $L_5$ libration point are located. To gather insight concerning the returns, the sphere objects representing the returns to the map are scaled based on the crossing number, $c_n$, which defines the accumulated number of returns. With this $c_n$ scaling, smaller spheres correspond to the first few crossings of the hyperplane $\Sigma_2$, whereas, larger spheres imply a crossing preceded by many previous returns. The resulting visualization, displayed in Figure 14(c), highlights two groups or clusters of orange and green crossings. Each group of crossings possesses the same color (i.e., originates from the same initial condition) and collects spheres of various sizes in the same general locations. Employing user selection for the green set of clusters reveals an out-of-plane orbit structure near $L_5$ that exhibits behavior similar to a quasi-periodic orbit for the specified time interval (Figure 14(c)). However, the green trajectory is not truly quasi-periodic since this orbital structure is not explicitly seeded inside the center subspace corresponding to a nearby periodic orbit. However, the green trajectory does remain in the vicinity of $L_5$ for the simulation length ($\approx 434$ days) and may remain in this vicinity longer with station keeping. With a visual analytics trajectory design approach, though, trajectories selected from a Poincaré section are immediately available and can be implemented as the focus of further analysis. (Reference 16 conveys additional sample hyperplanes and section visualization options.)

![Figure 14](image-url)

**Figure 14.** The two-sided Poincaré map for the hyperplane $\Sigma_2$ that intersects $L_5$ in the E-M system ($C = 2.96$).
SUMMARY AND CONCLUSIONS

The application of visual analytics to trajectory development demonstrates the interactive capabilities and the rapid design capabilities that are accessible in a visual environment. Interactive tools can also enhance automated design processes. Trajectory design within a visual scene or graphical user interface (GUI) expedites the design process when compared to segregated operations such as the manual manipulation of data and targeting schemes operated outside of a visualization component. When the governing differential equations are not solvable analytically, these capabilities can be an enabling technology. Expanding these functionalities into other design applications offers opportunities for rapid trajectory development and information discovery with various dynamical models – all via a visual environment.

A comprehensive interface for designing spacecraft trajectories that applies the various facets of the visual analytics process is introduced. Designs can originate with just small amounts of information, and all subsequent steps can be implemented visually in real-time to plan entire end-to-end trajectories. Unique user insight is also incorporated. Every step towards a design objective occurs in a single sitting through user-interactions with visualizations, data structures, and targeting algorithms. In the application to the CR3BP, the visual analytics design strategy requires a user with more experience in dynamical systems, but the results are generated exceptionally fast with a significantly reduced level of manual input.

The implementation of Poincaré maps in a real-time design setting is also explored for multi-body scenarios. The parallel computational power of the GPU is exploited to generate Poincaré sections for 2D grids of initial states in seconds, yielding map results that are available for almost instant analysis. Interactive transformation editors allow a hyperplane to be defined at any orientation and location; although the examples have all employed hyperplanes in configuration space, the concept is not limited to physical space. Interpretation of the information available in a map is aided by exploiting graphical options such as scale and color and by incorporating pick-callbacks to instantly view trajectories from the map.

With this application of visual analytics, extensions to this are available for further investigation. One element that greatly increases the appeal of trajectory design with visual analytics is an interactive and adaptive process for applying constraints on trajectory arcs. Expanding the applications of Poincaré maps and surfaces of section is also a priority. The results in this investigation display a real-time implementation of Poincaré sections, but there are many more functionalities to be explored. An extension to higher-fidelity models, e.g., time-varying and ephemeris models, is a continuing effort which, in turn, requires exploration of higher-fidelity numerical propagation on GPU architectures.

ACKNOWLEDGMENT

The authors are extremely grateful to Rune and Barbara Eliasen for their support of this research and funding for the Rune and Barbara Eliasen Visualization Laboratory at Purdue University. Also, the authors wish to acknowledge Visualization Sciences Group (the developers of Avizo® ) for programming and implementation assistance with the interactive and visualization tools in this work. This effort is also supported by the College of Engineering and the School of Aeronautics and Astronautics at Purdue University.

REFERENCES
