Advanced, Automatic Stream Surface Seeding and Filtering

Matt Edmunds, Tony McLoughlin, Robert S. Laramee, Guoning Chen, Eugene Zhang, and Nelson Max

Fig. 1. Automatic Stream Surface Seeding: Silhouette Edge Highlighting, Semi-transparent Surfaces implemented with Depth Peaking, and Curl-based filtering.

Abstract—The visualisation of 3D flow poses many challenges. Difficulties can stem from attempting to capture all flow features, the speed of computation, and spatial perception. Streamlines and stream surfaces are standard tools for visualising 3D flow. Although a variety of automatic seeding approaches have been proposed for streamlines, little work has been presented for stream surfaces. We present a novel automatic approach to seeding and filtering stream surfaces in 3D flow fields. The automatic algorithm captures the important characteristics of the flow by allowing the user to specify density of surface seeding. We describe defining and optimizing seeding curves at the domain boundaries from isolines generated from a derived scalar field. We detail the generation of an initial set of stream surfaces integrated through the flow and discuss the associated challenges of surface termination. We then present an algorithm that automatically seeds new stream surfaces at a user specified distance away from a given surface. The results demonstrate how we achieve satisfactory domain coverage and capture the features of the flow field. Strategies for resolving occlusion resulting from seeding multiple surfaces based on illustrative rendering and stream surface filtering are also presented and analysed.

1 INTRODUCTION AND MOTIVATION

Flow visualisation is a powerful means for exploring, analysing and communicating simulation or experimental results. The topic of flow visualisation using stream surfaces has become an increasingly important and popular direction of research in recent years.

Streamlines are an intuitive, fast, and simple method for visualising flow. Streamlines require less computation than surfaces and are generally easier to implement than their surface counterparts. These types of curves can present disadvantages, however, such as visual clutter (when too many streamlines are rendered) and lack of depth perception. Surface primitives (as opposed to curves) have well defined normals. Thus they offer perceptual advantages including: lighting and shading which provide intuitive depth cues, the ability to texture map including texture advection [14], the placement of additional geometry on the surface [18], and their use for depicting boundaries. Surfaces generally suffer less from visual clutter than lines, points, or other geometric primitives because they offer greater spatial continuity. Stream surfaces partition the flow domain into regions of similar flow behaviour. The same cannot be said of stream lines in 3D flow fields.

Stream surfaces for visualisation face many challenges. These surfaces must represent an accurate approximation of the underlying simulation. Adequate sampling must be maintained while reducing the unnecessary computational overhead associated with over-sampling. When using surfaces the problem of occlusion arises. This may stem from multiple surfaces that occlude one another, a large surface that results in self occlusion, or a combination of both. A general solution to this problem is to use transparency. With integral surfaces we have additional options. Illustrative techniques can be used to improve perception. Also stream surface seeding positions may be modified to reduce clutter.

Manual seeding is the most common method for the placement of stream surfaces. However interactive stream surface placement is based on trial and error. Important characteristics of the flow can easily be missed. Stream surfaces must be seeded such that they capture the features of the flow. A large body of research has been invested into automatic seeding strategies using streamlines, but, little has been offered for automatic stream surface seeding. This provides strong motivation for studying stream surfaces and their seeding. The main benefits and contributions of this paper are:

- A novel, automatic approach to seeding stream surfaces in 3D flow fields.
- The prioritisation of seeding curves generated at the boundary to produce superior and consistent visualisations.
- Seeding a complete set of surfaces on the domain boundary achieving adequate initial domain coverage.

\*\*\*\*\*\*\*\*\*\*

- Matt Edmunds, Tony McLoughlin, and Robert S. Laramee are with Swansea University, UK. E-mail: (consult, cation, r.s.laramee)@swansea.ac.uk.
- Guoning Chen is with University of Utah, US. E-mail: chenng@sci.utah.edu.
- Eugene Zhang is with Oregon State University, US. E-mail: zhang@eeecs.oregonstate.edu.
- Nelson Max is with University of California, Davis, US. E-mail: max@cs.ucdavis.edu.

\*\*\*\*\*\*\*\*\*\*

Manuscript received 31 March 2010; Accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.

For information on obtaining reprints of this article, please send email to: tvg@computer.org.
Fig. 2. A set of stream surfaces visualising simulation of flow behind a cuboid [9]. Transparency is mapped to surface curvature. An initial set of seeding curves can be seen on the far plane of the domain.

- New techniques for automatically seeding a new surface from an existing surface to address coverage in areas such as periodic orbits.
- Techniques for reducing occlusion related to seeding multiple surfaces based on filtering and illustrative rendering.

We illustrate how to achieve adequate coverage of the domain and capture the properties of the flow field. The key to capturing the characteristics of the flow is allowing the user to specify the density of surfaces at boundary and interior locations. Our focus pays particular attention to the seeding curve generation, inter-surface awareness employing distance fields, and occlusion.

First a review of related literature is conducted in section 2. Then a detailed presentation of the algorithm is given in section 3. The results are discussed in section 4. Conclusions and proposed future work are mentioned in section 5.

2 Related Work

This section studies the different techniques and contributions related to seeding integral objects in 3D.

A streamline is a curve that is everywhere tangent to a vector field. A streamline can be described as the union of all streamlines passing through a seeding curve. It can be approximated by generating a series of streamlines along a seeding curve and joining them to produce a polygonal representation. Stream surfaces are useful for understanding flow structures within a single time step or static flow field.

2.1 Seeding

Van Wijk introduces a novel method for the construction of stream surfaces [28]. This paper provides an alternative stream surface construction technique [based on the global representation of the stream surfaces as implicit surfaces $f(s) = C$]. The Van Wijk method constructs surfaces only where intersections with the boundaries take place.

Zöckler et al. introduce a method of illuminating streamlines [34]. There is no native support for the lighting of line primitives in graphics libraries such as OpenGl due to the fact that line primitives have no unique normal vector. A streamline placement algorithm has been introduced. For the placement technique a stochastic seeding algorithm is applied. The degree of interest in each cell is defined by some scalar value (i.e., velocity magnitude). See Weinkauf et al. [32] [31] for applications of this seeding strategy.

Cai and Heng present [2] the principal stream surface algorithm, which automatically generates stream surfaces that properly depict the topology of an irrotational flow. The principal stream function describes a scalar field representing the direction of velocity, and is visualized using volume rendering.

Mattausch et al. [22] combine the illuminated streamlines technique of [34] with an extension of the evenly-spaced streamlines seeding strategy of Jobard and Lefer [9] to 3D.

Theisel et al.'s approach to constructing saddle connectors in place of separating stream surfaces is an effort to address the challenges of occlusion [27]. This endeavour was extended by Weinkauf et al. [33] using separating surfaces and connectors originating from boundary switch curves. The approach by Vilanova et al. [29] concentrates on seeding hyper-streamlines and the use of surfaces to visualise Diffusion Tensor Imaging data.

Chen et al. [4] present a novel method for the placement of streamlines that does not rely solely on density placement or feature extraction. This approach is based on a similarity method which compares candidate streamlines based on their shape and direction as well as their Euclidean distance from one another.

Li et al. [16] present a streamline placement strategy for 3D vector fields. This is only applicable if the kind where an image-based seeding strategy is used for 3D vector visualisation. Interactive seeding strategies have been used in various modern, real-world applications including the investigation and visualisation of engines simulation data [11] [12] [13].

An image-space-based method for placement of evenly-spaced streamlines on boundary surfaces is presented by Spencer et al. [26]. This method is projected onto the image plane. Thus, the complexity of tracing in the large unstructured grids that typically result from CFD simulations is avoided. Streamline density is controlled by an adaption of the method of [9].

Peikert et al. [24] present topology relevant methods for constructing seeding curves, which producing topology relevant stream surfaces, to visualised critical points and periodic orbits.

Forstl et al. [6] introduce real time construction and rendering of separation surfaces in conjunction with the rendering of the streak surface particles.

More recently Marchesini et al. [20] present a view-dependent strategy for seeding streamlines in 3D vector fields. No distribution of streamlines is ideal for all viewpoints. Therefore, this method produces a set of streamlines tailored to the current viewpoint.

Edmunds et al. [21] present a work in progress short paper describing a surface seeding method at the domain boundary. This work describes the construction seeding curves based on the scalar of flow exit trajectory. This work lacks full domain coverage as only one iso-value is used for the seeding curve construction. This leaves areas of the domain un-visualised. Also areas of the domain where flow cannot be traced to the boundary are not visualised. The other limitations of this work include: Algorithm parameters that require further analysis e.g., the choice of isovalues and initial seeding curves need more exploration, rendering surfaces with transparency require more pre-processing, further analysis-surface filtering needed to be conducted, a study of boundary seeding curve prioritisation, a study of the technique for terminating surfaces, and control over the density of surface seeding.

2.2 Surface Rendering and Visualisation

Löffelmann et al. [19] introduce methods for placing arrows on stream surfaces which improve perception of flow. The offset separated portions of the surface alleviate occlusion. Löffelmann et al. [17] improve on this using hierarchical techniques to better fit the surface tiling. Lamee et al. [14] draw on previous image-based texture-advection research in order to improve the information content and perception of flow on stream surfaces.

Progress for the illustration of stream surfaces has been made by Born et al. [1] and Hummel et al. [8]. These algorithms concentrate on illustrative renderings of the surfaces while maintaining interaction.

2.3 Title

While many papers have been published presenting detailed algorithms for the construction of stream surfaces, extending the work by Edmunds et al. [21] this is the first paper (to our knowledge) to present an automatic seeding algorithm for stream surfaces which fills the domain.

Space limitations prevent an overview of stream surface construction algorithms. For a complete overview of streamline seeding strate-
Seeding Curve Generation
Generate Spline Field
Surface Generation
Render
Statistical Technique Processing
Check for Termination
Seeding Curve Generation
Select Next Unsearched Surface
Surface Generation
Render
Statistical Technique Processing
Check for Termination
(b) The interior seeding curve and stream surface pipeline. Each surface, in a list of existing unsearched surfaces, is searched along its length to locate a point at which the closest distance to each discretised timeline vertex, and then analysed for suitability against a predefined set of parameters. If the result curve does not pass the criterion then the next timeline in the surface is then selected and the process starts again. Section 3.3 provides details.

3 AUTOMATIC SURFACE SEEDING

This section presents the automatic stream surface seeding algorithm, starting with an overview of the seeding pipeline illustrated in figure 3(a). This algorithm is described in several stages; initial seeding curve generation and prioritisation, initial surface computation, seeding curve generation from existing surfaces, surface filtering and rendering. The following sections include a novel algorithm for generating seeding curves from timelines offset at a preset distance using the surface normals. The achievement of complete domain coverage in areas such as periodic orbits (recirculating flow), is the focus of the seeding strategies presented in this paper. Generation of surfaces in areas of flow which cannot be traced back to the domain boundary is important in order to visualise the full set of flow characteristics. The interior stream surfaces algorithm achieves coverage in areas otherwise empty after the boundary seeding process is complete. The proposed recursive pipeline is split into three main stages: Timeline based seeding curve generation and criterion testing, interior curve refinement, interior surface generation and rendering. An overview of the algorithm pipeline is illustrated in figure 3(b).

1. The starting point is the derivation of boundary seeding curves. This is realised by generating the seeding curves from isolines derived at the domain boundary. A scalar field is derived at the domain boundary based on the direction of flow exiting the domain. The isolines are constructed using a marching squares algorithm. A simple point sorting technique is employed to store the vertices in the required order. Once the vertices are correctly stored, the seeding curves are then prioritised into the correct seeding order. Refer to section 3.1 for details.

2. Once the boundary seeding curves are computed, the next step generates a collection of stream surfaces which advance through the vector field until they meet a set of terminating conditions. The termination parameters include maximum streamline length and distance to neighbouring surfaces. The minimum distance between each surface is calculated using a distance field. Each surface is then terminated. The process is repeated from section one until all isosurfaces are seeded. When all surfaces are completed the algorithm proceeds to the next step. Section 3.2 and 3.3 provide details.

3. This step involves searching the initial list of unsearched surfaces seeded from the domain boundary along their length to find empty regions at which seeding of new surfaces can take place. Each timeline of a given surface is offset normal to each discretised timeline vertex, and then analysed for suitability against a predefined set of parameters. If the result curve does not pass the criterion then the next timeline in the surface is then selected and the process starts again. Section 3.4 provides details.

4. If the interior candidate seeding curve passes specified criterion, the next step is to produce a smooth and ordered seeding curve for generating a new surface. This refinement is achieved by removing vertices which are in close proximity to each other and interpolating new vertices to maintain an evenly distributed smooth discretised curve. Section 3.5 provides details.

5. Once a suitable location to generate a new surface has been found, the originating surface is transferred to the processed list and searched for no further. The new interior stream surface is added to the list of unseeded surfaces. The surfaces are searched on both sides, and recursively through the list of unseeded surfaces until the list is empty. The final step in the pipeline is to render the scene. The techniques used here are consistent with boundary seeding. Section 3.6 provides details.

6. The final step in the pipeline is to render the scene. A number of techniques are implemented to reduce occlusion and aid the viewer in perceiving the resulting visualisation. This includes the use of transparency, colour, clipping planes, edge highlighting, lighting and shadow, and surface filtering. See section 3.7.

3.1 Boundary Seeding Curve Generation

Generation of seeding curves from isolines derived at the domain boundary is performed in three steps. The first step is to define a scalar field based on exit flow at the domain boundary. The scalar represents the angle of incidence between the vector field defined at domain boundaries and the domain extent itself. The calculation is performed by projecting the unit vector onto the domain boundary. The resultant magnitude is used as the scalar. If the exit trajectory is perpendicular to the domain boundary a scalar value of zero is stored, if the exit trajectory is parallel to the boundary then the scalar is stored as unity.

The next step is to construct the isolines from the scalar field at the boundary using a marching squares algorithm. The resulting vertices would normally be rendered as order independent line segments. However the vertices require correct ordering for the seeding curve.
Modifying the initial isovalue used for the generation of the seeding curves produces results that are different. A general isovalue of 0.5 was found to be adequate to produce reasonable visualisations. The main point of using isolines derived from exit flow direction is the binding of the coherent flow structures at the boundary and tracing them through the domain. Boundary switch curves [33] are an example of this, where an isovalue of 1.0 is equal to a boundary switch curve.

The range of isovalue used in the construction of the isolines, and resultant density of seeded surfaces, can be specified by the user. This effectively seeds surfaces between pairs of boundary switch curves, where they exist. The algorithm subdivides the range between zero and one then back again. The seeding curves, and then surface generation is performed from the last isovalue to the last generating the surfaces during each loop.

The seeding curves are prioritised in order of surface generation. The order of seeding new surfaces can influence the resulting visualisation. A logical order is linear seeding curve first. This heuristic is based on the idea of producing large domain filling surfaces first. After some experimentation we find that seeding from closed loop curves is beneficial to visualising vortex cores. The final heuristic is to seed closed loop curves first, large seeded, and then seed open ended curves; large to small, starting with an iso value nearest one, seeding each set of isolines to the smallest isovalue.

3.2 Boundary Surface Generation

Stream surfaces (our work uses Garth et al. [7] as an out of the box implementation) are propagated from each of the seeding curves defined in section 3.1. The surfaces are then terminated according to a set of terminating conditions. Maximum surface length, boundary proximity, and distance to neighbouring surfaces are used to determine termination.

Calculating surface length and determining boundary proximity are straightforward. However a distance field is used for the efficient detection of neighbouring surfaces. As each surface is generated, its location is added to the distance field. Then the field is updated. As the next surface is propagated through the domain, it is tested against the distance field to determine if the proximity to any neighbouring surfaces is less than a predefined minimum distance. If so the surface propagation is terminated. This process is repeated for all surfaces.

After the initial set of stream surfaces is constructed from the domain boundaries, additional stream surfaces can be seeded from existing ones by a user-defined separating distance in order to gain complete domain coverage.

3.3 Distance Field

The detection and computation of the distance to closest object in a given domain from any given point is non-trivial. A brute force approach can be implemented computing the distance of every vertex of every object in the domain, with the current vertex and storing the shortest. This method is very expensive and thus distance field techniques to improve the speed are reviewed.

In general, there are two groups of approaches [10]: Distance computation for common surface representations, which keeps the above basic scheme but discards most of the objects by exploitation of spatial coherence e.g., computing distances from objects in close proximity. Distance transforms which initially use a similar technique to evaluate distances to certain regions e.g., thin layer around the surface, and then propagates them through the whole volume.

The most interesting method appropriate for use with discretised surfaces, is the Vector City Distance Transform or VCDT [25].

The first step is to calculate the distance from the vertex of interest (in our case surface vertices) to the bounding cell vertices 4(a). These vertices are compared with the currently stored vertices at these locations and the shortest stored 4(b). Once this is done for every vertex of the new surface, the distance field is then propagated throughout the domain 4(c). When the next surface is computed, each vertex is added to the bounding cell’s vertices in turn and the shortest result is then compared to the desired minimum 4(d). If the distance is too short then the surface front propagation is terminated.

3.4 Timeline-Based Seeding Curve Generation and Criterion Testing

The pipeline takes as input a list of stream surfaces seeded from the domain boundary from which additional surfaces can be seeded from by a user-defined separating distance in order to gain complete domain coverage. This list is initially unseeded.

Each unsearched surface is searched in turn to find a suitable location to generate a new seeding curve. This step involves offsetting each timeline along the length of the surface to find empty regions at which seeding of new interior stream surfaces can take place.

Each timeline of a given stream surface is projected along the surface normals, and analysed for suitability against a predefined set of parameters. As the surfaces are searched both along their front and back faces, a set of vertices are also projected from the timeline in the reverse normal direction.

The first criterion which the candidate seeding curve must pass is the boundary test. Every new vertex of the candidate seeding curve must be inside the domain boundary. Therefore any vertex projected outside the domain is marked as reject. Refer to Figure 5(a).

The second criteria which must be passed is the distance test. For this test each new vertex is tested against the distance field. The vertex is marked reject if the location is within a user defined minimum distance parameter e.g., too close to a neighbouring surface. Refer to Figure 5(b).

The new curve is then tested for spatial continuity guided by user defined parameters of minimum contiguous length, and maximum split distance. If any resulting gaps in the offset timeline are too large, or any remaining section is too short, then the complete candidate seeding curve is then rejected.

If the resultant curve does not pass the criteria then the next timeline along the length of the surface is then searched, and the process starts again.

3.5 Interior Seeding Curve Refinement

If the interior seeding curve passes the previously described criteria, it is then refined. The refinement process is intended to smooth out any inconsistencies with the new curve, and evenly space the resultant
vertices. As a result of projecting the vertices along surface normals in curved areas, concave or convex, the vertices may be too close to one another, too far apart or even change their order along the curve.

The refinement starts by removing vertices in too close proximity. The vertices are tested in groups of three starting from one end of the curve, incrementing one vertex at a time. The purpose of this approach is to test the central vertex against a user defined proximity to its neighbours. If a neighbouring vertex is too close, it is removed. Refer to figure 5(c).

The curve is further refined by inserting additional vertices at locations where proximity to the next vertex is too great. The insertion process uses cubic interpolation (Catmull Rom spline). The process of insertion starts with marking the position along the curve a new vertex should be inserted, a new vertex is interpolated and held in a temporary list. This is repeated for every section along the curve. Refer to figure 5(d).

Once finished the temporary list of vertices is then inserted at the correct locations along the curve. The approach of inserting the vertices into the array post interpolation, prevents newly inserted vertices from violating the proximity test, and interfering with the interpolation. The insertion stage of the pipeline recursively inserts new points until every segment meets a minimum user defined proximity test. Refer to figure 7(a) and 7(b).

3.6 Interior Stream Surface Generation

Once the seeding curve has been formed, a new surface is generated in both forward and reverse directions. The distance field is updated with the vertices representing the new surface. The given stream surface is added to the list of processed surfaces, while the new surface is added to the list of unprocessed surfaces. This is repeated for every unprocessed surface until no further seeding curves can be generated.

Calculating surface length and determining boundary proximity are straightforward. However a distance field is used for the efficient detection of neighbouring surfaces. As each surface is generated, its location is added to the distance field. Then the field is updated. As the next surface is propagated through the domain, it is tested against the distance field to determine if the proximity to any neighbouring surfaces is less than a user defined minimum distance. If so the surface propagation is terminated. This process is repeated for all surfaces.

3.7 Stream Surface Filtering and Rendering

Rendering of the scene is the final step in the pipeline. A number of techniques are implemented to aid the viewer’s perception of the resulting visualisation and to aid in the reduction of occlusion. The techniques used to represent the results include the use of transparency, colour, silhouette edge highlighting, lighting and shadow, and surface filtering.

Colour is used to represent velocity magnitude, while transparency is mapped to vector field curl. Lighting and shading are standard tools to aid in the perception of depth, and shape. Silhouette edge highlighting is used to help the viewer understand where the surfaces curve away from the viewer, and enhance the perception of surface edges.

Another technique involves filtering of the surfaces to aid in the reduction of visual clutter. This technique involves filtering out pixels or fragments within the graphics API pipeline which meet user specified criterion. Our visualisations are rendered with transparency mapped to the magnitude of vector field curl calculated at each grid cell location. The result is a visualisation which is opaque in areas of high vector field curvature e.g. vortex cores, and transparent in regions of steady flow. Our filtering technique removes fragments which fall below the specified criterion. This criterion is an adjustable normalised scale which can interactively be adjusted by the user. Refer to figure 8.
4 Results

We achieve satisfactory domain coverage and capture the properties of the flow field by allowing the user to control the density of neighboring stream surfaces. Figure 6 shows results from seeding tornado data. It can be seen that the domain is adequately seeded to capture the structure of the tornado. This particular example does not demonstrate full domain coverage. Using translucency and silhouette edges improves the users perception of the results. This combined with filtering some of the generated surfaces aids in reducing the occlusion.

Figure 2 demonstrates capturing the vortices generated behind the back face of the cuboid (cuboid not rendered for clarity). Used in conjunction with translucency the perception of the vortices are enhanced.

In figure 9 the seeding of the surfaces fills the domain, capturing the features of the flow. The complex flow structures are well represented with our technique. Some of the seeded surfaces are again filtered in order to reduce occlusion and visual complexity, this is further enhanced by using transparency to visualise important parts of the data otherwise occluded.

The illustrative strategies implemented for resolving occlusion resulting from seeding multiple surfaces improve perception and therefore aid understanding of the underlying flow structures. Visualisation of less complex flow characteristics such as figure 6, produce clear results from the different strategies employed.

When visualising more complex flow data such as the Bernard flow simulation figure 9, the issue of occlusion can significantly increase. The ability of the user to be able to filter specified surfaces from the rendering can reduce much of the clutter improving the overall visualisation.

The seeding strategy employed removes the need for the user to conduct lengthy examinations of the flow fields using manual seed placement techniques. The technique shows adequate domain coverage, and captures the features within the flow field for all the datasets we experimented with.

We have tested our algorithm on a variety of simulations, ranging from simple to complex including the simulation of a tornado, flow past a cuboid, Bernard flow as well as others. We found our visualisations were consistent with previous work and captured the same features [15] [5] [30].

5 Conclusions and Summary

We introduce a novel automatic method for the seeding of stream surfaces, and investigate a range of methods for the reduction of occlusion.

Despite the great amount of progress that has been made in the field of flow visualisation over the last two decades, a number of challenges remain. Challenges such as automatic path surface placement and perception remain key topics for further research.

Extending this algorithm to time-dependant flow fields is a topic for further research.

References


5-8: Comments from review on 15 March 2011
9-10: Seeding curves and their singular surfaces.
11: Define prioritization priority
12: (I would recommend American English as this is a journal based in the US. There is a bias.)
13: Capture the properties of the flow including areas of recirculation
14: Based on distance fields
15: Based
16: Leave
17: (No new)
18: The main focus of this work is on accurate surface construction rather than on a specific algorithm.
19: (How is this different from this paper? More detail.)
20: Cops in domain coverage.
21: User-interactive manual interaction
22: Is based on user input
23: Surfaces are not filtered in any way
24: All boundary seeding curves are treated equally
25: No description of how distance fields are used is given
26: And surfaces cannot be seeded from existing surfaces.
27: Optimize
28: Combine these two pipelines into one, don't be afraid if the diagram spans both columns. This is an algorithm from start-to-finish.
29: Updated
30: Interior
31: Composite
32: Iterative
33: In the direction normal to the surface
I think this needs to be updated.

Can you add context from (33) stating why boundary switches curves are interesting or useful?

What does this mean? Any why?

Iteratively

Conjecture

Conjecture to sharpen

In increasing order of

Implement

For surface construction such as

Testing

(i.e., lower case)

Are a second category

Give this vertex a label, such as $p_i$, and then refer to it using this label in both the text and the figure.

Again, it's best to provide explicit examples of using vertices with labels, e.g., $p_{01} - p_{07}$, and use most labels to refer to the vertices. See Tony's sensitivity paper for good examples of this.
\[ p_i, p_{i+1}, p_{i+2} \]

\[ p_0 \]

\[ p_0 \text{ and } p_0 \text{ in this example complete} \]

\[ \| p_i - p_{i+1} \| < \alpha_{\text{min}} \] (for all of these tests, you'll have to express them mathematically.)

define this mathematically.

visualizes

error prone

you'll have to include some analysis of performance time