

Stereo-consistent Contours in Object Space

Dennis R. Bukenberger, Katharina Schwarz, Hendrik P. A. Lensch

Department of Computer Graphics, Eberhard Karls University, Tübingen, Germany

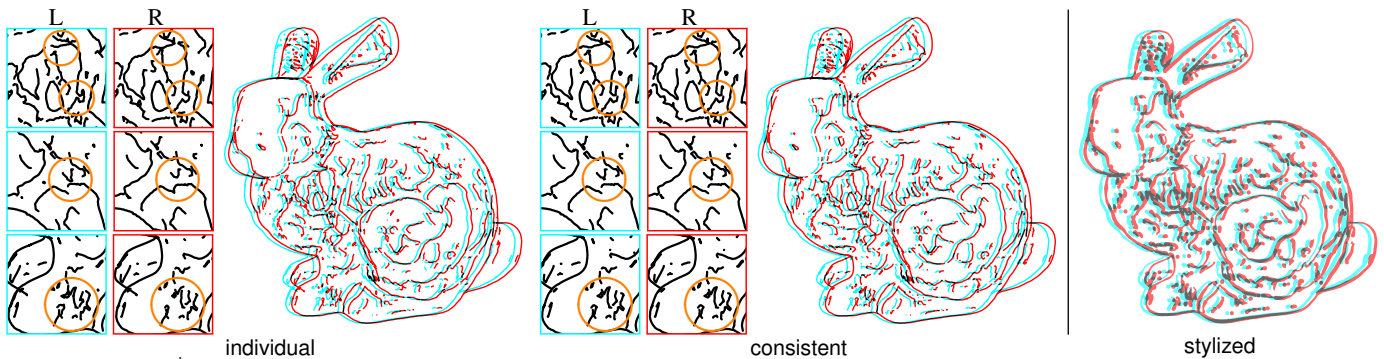



Figure 1:  ‡ Individually generated contour lines (left) cause viewing discomfort due to binocular rivalry. In the stereo-consistent results of our pipeline (middle, right) these rivalries are removed.

‡ Figures marked with this icon are best examined on-screen with red(left)-cyan(right) anaglyph glasses.

Abstract

Notebook scribbles, art or technical illustrations - line drawings are a simplistic method to visually communicate information. Automated line drawings often originate from virtual 3D models, but one cannot trivially experience their three-dimensionality. This paper introduces a novel concept to produce stereo-consistent line drawings of virtual 3D objects. Some contour lines do not only depend on an objects geometry but also on the position of the observer. To accomplish consistency between multiple view positions, our approach exploits geometrical characteristics of 3D surfaces in object space. Established techniques for stereo-consistent line drawings operate on rendered pixel images. In contrast, our pipeline operates in object space using vector geometry, which yields many advantages: The position of the final viewpoint(s) is flexible within a certain window even after the contour generation, e.g., a stereoscopic image pair is only one possible application. Such windows can be concatenated to simulate contours observed from an arbitrary camera path. Various types of popular contour generators can be handled equivalently, occlusions are natively supported and stylization based on geometry characteristics is also easily possible.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Visible line/surface algorithms

1. Introduction

Line drawings are an easy way to depict three-dimensional shape information in a planar illustration. However, lines that originate from 3D objects actually exist in the object's 3D space. Due to the projection into 2D image space, their depth components are omitted and one can no longer experience their three-dimensional nature. In this work we analyze how and the extent to which the boundaries of three-dimensionality can be stretched for line drawings. Lines that are bound to an object's geometry are not the problem because they are consistent for multiple view positions. The challenging part is to obtain the consistency of lines which depend on the relative viewing position of the observer. These view-dependent lines have to be consistent for both eyes to avoid binocular rivalry and viewing

discomfort (Examples are illustrated in Figure 1). We propose a solution that operates in object space and utilizes three-dimensional geometric properties of the surfaces to maintain consistency for a generalized multiview application. Epipolar constraints can be met in a specialized case for stereo-vision.

Many publications have faced the difficulties of stereo-consistency in NPR. However, these approaches often focus on stereo-consistent stylization of stereoscopic images or videos and, therefore, the proposed methods were developed for the image space. Kim et al. [KLKL13] especially focused on line drawings from 3D objects but their solution for stereo-consistency is again only an image space approach. Nevertheless, they are able to create line drawings which are stereo-consistent for the left and right eye.

The core idea in our work is to approach this task directly in three-dimensional space. This allows us to exploit geometric characteristics of the 3D surfaces from where the contour lines originate. Furthermore, our concept is developed for a generalized multiview window. Projections of a deformed center-view contour can seamlessly interpolate between different view positions in this window. Stereoscopic results for left and right eye can easily be generated by sampling the multiview window from two points on a horizontal line. Correct contour visibility is established with a view graph algorithm which takes pointwise visibility samples as input. For robustness we incorporate local geometry interpolation with Bézier curves which improves ray-test results. Ray-tests are evaluated only for the center and are re-established consistently for all other views. Another benefit of this concept is that view-dependent occlusions are natively incorporated. Moreover, our approach allows for temporal- and stereo-consistent contours in a multiview window or on arbitrary fly-around camera paths.

1.1. Overview

Figure 2 illustrates schematic details of our proposed pipeline, which consists of the following four main parts.

I. Initialization: The camera and scene are initialized when mesh data of selected models is loaded from file. View-independent auxiliary information like surface curvature is computed with standard algorithms, this part is not further elaborated with technical details.

II. Contours: Lines on objects are extracted by selected contour generators. Dependent on the camera definition, this step is repeated for specified viewpoints, e.g., **Center, Left, Right, Up** and **Down** for a multiview camera. Visibility of contour points is sampled with ray-tests but only for the master camera. The different contour sets are matched and combined to one super-structure.

III. Interpolation: An arbitrary camera position from the multiview-camera is instantiated. The superimposed contour interpolates view-aligned contours for the chosen camera position.

IV. Visibility: View aligned contours are linked together in a specialized graph structure. This view graph implements algorithms to reestablish correct view-dependent visibility and occlusion. Eventually the view graph is projected into the image plane resulting in a vector graphics output.

1.2. Related Work

Line drawings: Three-dimensional objects can be depicted with various kinds of feature lines [CGL*08, RCDF08]. E.g., construction blueprints often feature creases to denote sharp edges of an object. Line drawings of organic or curved objects may use ridges and valleys [OBS04] or crest lines [YBS05] to emphasize shape. If surfaces of virtual 3D models penetrate each other, the resulting intersections [Mö197, GD03] also provide important cues on the surfaces' form. So far, all these lines were solely dependent on the objects' geometry and nothing else. However, the more relevant lines for our work also depend on the relative position of the observer like the silhouette contour [HZ00]. *Suggestive contours for conveying shape* [DFRS03b] are lines which would be silhouette contours from nearby viewpoints. As Decarlo et al. [DFRS03a] state,

their suggestive contour generator can create quite pretty pictures but also some not so pretty ones. They justify this statement with the complexity of involved operations for the computation of curvature. Especially because of curvature itself, its second and third derivative are quite sensitive to noise and under-tessellation. Artifacts from noise can show up as wrinkled lines on otherwise smooth surface areas. In our results, contour lines or visible segments are not filtered to have a minimum length. This is why results of other implementations [DFRS03a] may appear *cleaner*. View-dependent curvature is used to compute apparent ridges [JDA07] which extend traditional ridges with view-dependency. Described contour generators usually extract lines from objects given as triangulated meshes. Bénard et al. [BHK14] proposed to improve contour generation with accurate topology. Our pipeline features view-dependent silhouette and suggestive contours as well as view-independent creases. Additional contour generators can be added easily.

Stereo-consistent stylization: Various publications investigate stereoscopy in combination with nonphotorealistic rendering (NPR). Richardt et al. [RSD11] proposed a coherence-based computational model to investigate viewing discomfort from independently stylized left and right images. An adaptation of Hertzmann's curved brush strokes [Her98] for stereoscopic content was proposed by Stavrakis and Gelautz [SG05]. Warping strokes from the left to the right image already improved stereo-consistency compared to naive approaches, but still had its flaws. Northam et al. proposed a novel approach [NAK12, NAK13], using discretized depth layers to extract coherent image regions for stylization. Layers of depth-aware strokes are merged accordingly to create stereo-consistent stylizations of a left and right image. However, these methods mainly focus on image stylization often using stereoscopic images or videos as input. An established image space approach for stereoscopic line drawings proposed by Kim et al. [KLKL13] searches and labels corresponding pixels in rendered line drawings to establish stereo-consistency. The concept introduced by Liu et al. [LMY*13] is able to superimpose stereoscopic depth on simple monoscopic cel animations.

View-dependency and multiview: Rademacher's approach for view-dependent geometry [Rad99] captures view-dependent distortions of objects or characters which are not possible with ordinary 3D models. His proposed technique specifies warps of a base model which are interpolated between a set of key deformations based on a specified viewpoint. Visibility, occlusion and especially the transition between both are not trivial to determine for line drawings. Many publications [GGSC98, NM00, Mar00] refer to an ID-image approach which resolves visibility with labeled segments in image space. The method used in our implementation builds on line segments in object space analogous to Hertzmann and Zorin's [HZ00] and resolves ambiguities in a view graph as denoted by Bénard et al. [BHK14]. The concept of perspective projection is fundamental for many applications in computer graphics. Kooima [Koo09] discusses misassumptions which arise from a standard perspective camera model used in a stereoscopic or multiview context. Results of our pipeline are created using an adapted version of the skewed frustum perspective camera model to avoid inconsistent distortions for left and right view.

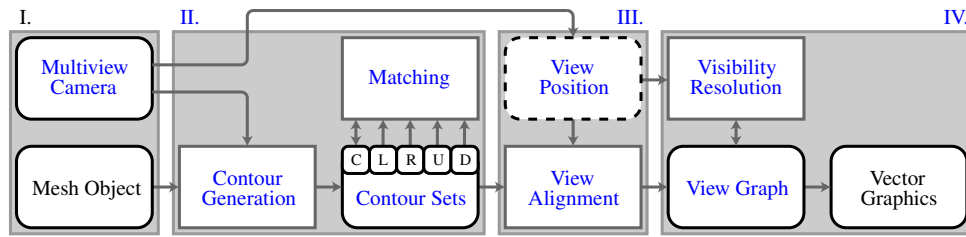


Figure 2: **I.** Initialization, **II.** Contours, **III.** Interpolation and **IV.** Visibility are the four main steps of our pipeline, as described in Section 1.1. Rounded boxes are instances, straight boxes are procedures. In the PDF, blue labels are clickable links to the associate section in the paper.

2. Motivation

This section gives an outlook on our anticipated goal, the challenges we had to face and how they were resolved.

2.1. Why multiview line drawings?

Technical details on multiview and contours are described in the following two sections. For now, multiview can be seen as the degree of freedom one has, sitting at a table with an object on it. Straightening up, ducking down or leaning to a side defines the extents of this multiview range, we call this the multiview window. The observer's eyes already sample the object from two separate positions in space but at the same time. Synthetically this can be approximated with an autostereoscopic display and head tracking. Fields of application like construction, medicine or art often favor line drawings over other visualizations due to an easily adjustable level of detail. However, line drawings are not trivially suitable to be displayed in a stereoscopic context or to be generated on the fly for real-time head-tracked positions. The goal of our work is to determine meaningful links between precomputable partial solutions in order to interpolate intermediate results. Therefore, our proposed interpolation technique is designed to guarantee valid approximations of interpolated points for all view positions. Furthermore, we intend to concatenate several multiview windows, which allows for simulating an arbitrary fly-around camera path.

2.2. Multiview terminology

A common understanding of stereoscopy refers to the concept of two images of identical content, taken with a small horizontal offset. In this work, the term multiview can be seen as a generalization of this idea with a whole array of images, also taken with vertical offset. As a real world scenario this could be realized with a camera array [LH96], a lightfield camera or with a moving camera taking pictures of a static scenery from different positions over time. In this synthetic scenario, however, multiview is not limited to a set of fixed discrete positions but can be expressed as a bounded continuous range for a variable center of projection, as illustrated in Figure 3. To produce an array of images, one would just need to sample this range from a set of desired positions. Considering the established understanding of a stereoscopic image as two samples from this continuous multiview plane, the terms monoscopic-multiview and stereoscopic-multiview may require some explanation. For *monoscopic-multiview* it would actually suffice to just

move one perspective camera in the multiview range: The captured content changes as the camera moves. These changes in shape and visibility of object features are not problematic because the view position moves continuously in space over time. For *stereoscopic-multiview*, however, camera positions are different in space but co-exist at the same time. A problem that arises for targeted perspective cameras in this stereo context is the keystone effect: Projections for both views feature different distortions of the same object. We use a skewed-frustum model [Koo09] and a fixed image plane for all viewpoints to create stereo-consistent projections.

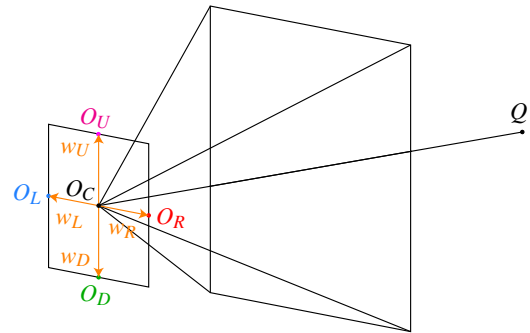


Figure 3: Vectors w_k ($k \in [L, R, U, D]$) span up the camera's multiview window. The window boundary points O_L, O_R, O_U, O_D and O_C are used to extract the initial sets of contours.

2.3. Contour lines and view-dependency

As there is a variety of lines one can draw to depict the 3D shape of an object [CGL*08], there are also various algorithms to extract such feature lines from virtual objects [RCDF08]. These algorithms can be loosely categorized as view-independent and view-dependent. Lines such as ridges, valleys and creases solely depend on an object's geometry, shape and curvature. However, apparent ridges, silhouette and suggestive contours change dependent on the viewing position. The latter group of view-dependent lines is of special interest in our work since our anticipated multiview camera literally defines multiple view positions. Therefore, contours have to become consistent in shape and at least in relative position when they are observed from different viewpoints. Why is this a desirable goal? First, if the view position moves in the multiview window with time-continuous motion, the contours on the object should neither suddenly nor drastically change. Furthermore, in a stereoscopic scenario the image content for both eyes must correlate. Binocular rivalry from inconsistent stereo images causes viewing discomfort and severely disturbs the overall depth impression.

2.4. Constraints to combine view-dependency and multiview

For the development of a stereo-consistent line drawing concept, we specified the following goals:

1. Resulting lines should be correct 2D projections of 3D points.
2. All views should feature the same set of contours.
3. Contours should not occur exclusively in single views.
4. The contour shape should be coherent across all views.
5. Topology should be coherent if it affects the contour shape.

We achieved these goals with a center-view concept by determining matches between contours across different views. Matched contours are merged to form a super-structure which is able to interpolate consistent contours for different viewpoints.

3. Contours and Matching

In this section we will address the contour generation and matching, grouped as part **II.** in our pipeline from Figure 2.

3.1. Contour Generators

After the mesh data is loaded, normals and curvature are computed. View-independent contour generators extract polylines solely based on mesh properties: Creases and sharp edges are selected by examining the enclosed angle between adjacent faces. Crests, ridges and valleys can be determined based on local gradients and algorithms similar to Canny edge detection. View-dependent contour generators like apparent ridges, suggestive and silhouette contours also require a specific viewpoint: The silhouette, for example, is defined as points where surface normal and view direction are perpendicular, so their dot product is zero. In our framework, contours are precomputed for a fixed minimal set of viewpoints from the multiview window, namely O_k ($k \in [C, L, R, U, D]$) as shown in Figure 3. These contour sets define the extents of the interpolation space, which follows in Section 4.

3.2. Contour Matching

The contour extracted from O_C is considered to be the master-contour. Pointwise matches are established from this contour's points to the points of other contour sets. To establish these matches we use a simple nearest neighbor approach with three-dimensional euclidean distance [LWM15]. This simple concept is able to find valid matches (Figure 4a) despite of different topology. If points happen to be falsely matched (Figure 4b), they are nonetheless close and presumably originate from the same surface. Thus, tangents of false matches are usually still sufficient for further usage.

We have considered other matching algorithms which incorporated more information on the polyline shape, topology and origin surface. Furthermore, we extended our polylines with snaxels [KH11] to act like active contours [KWT88] and then tracked their topological changes as they converged for a new view position. Leaving the tremendous overhead of some of these methods aside, none of them were able to outperform the simplest nearest neighbor solution in a general application scenario. Experiments with interpolation and alignment of polylines to determine bijective matches have also failed our expectations.

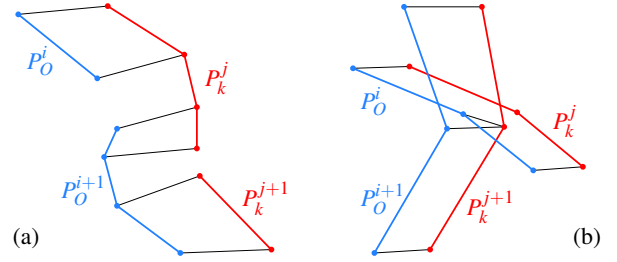


Figure 4: Pointwise nearest neighbor matching between Polylines from O_C and O_k contours. (a) Valid matches even with topological differences. (b) Failed nearest neighbor on crossing segments.

After the matching, the center-view contour is extended with the whereabouts of corresponding points from other contours. This augmentation is the basis for the upcoming interpolation step.

4. Multiview Alignment

Part **III.** of our pipeline from Figure 2 is called *view alignment* because now the previously augmented master contour is warped, so to say *aligned*, to fit a selected viewpoint. Stereo-consistency actually requires a tweak of the more general multiview concept. This is why we start with the general concept and introduce the specialized stereo-case as follow-up.

4.1. Tangent Plane Interpolation Concept

Now that correspondences between the master contour and points from other contours are established, this section will address our procedure to interpolate valid points in between. A simple linear interpolation would not only be incorrect, it would also be a waste of all the effort that was spent on analyzing the surface for contour generation itself. Figure 5 illustrates a stereo-setup as seen from above. p_L, p_R and p_C denote points of silhouette contours generated from different camera origins O_L, O_R and O_C respectively. Point pairs (p_C, p_L) and (p_C, p_R) were determined as described in Section 3.2 with point p_C as the master. To recapitulate the essential problem: p_C is not a correct contour point for any other viewpoint than O_C . p_C is occluded as seen from O_L and when observed from O_R , point p_C is somewhere on the object but not at the silhouette. We are looking for a line to interpolate between valid solutions.

The multiview window around O_C allows to express arbitrary viewpoints O_A within this window as simple linear combinations (Equation 1) of w_k and weights λ_k . The challenging part is to construct correct contours for an arbitrary viewpoint with the information we have: The center-view contour and the corresponding contours from the other viewpoints. However, we do not actually construct correct contours but exploit the fact that a correct projection is already sufficient for our final result. This is also illustrated in Figure 5: Point p_L is the correct contour point for O_L with projection q_L and p_C is the contour as seen from O_C with projection q_C . However, there is a point that creates valid projections for *both* viewpoints: Point p'_L is defined as the intersection of the view-tangents at p_L and p_C . The same holds analogously for O_R, p_R, q_R and p'_R .

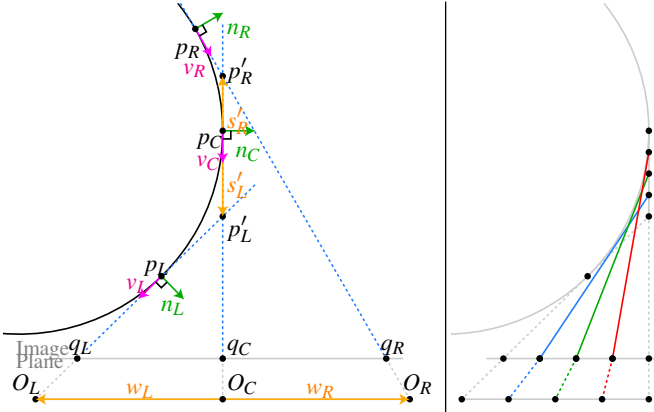


Figure 5: Left: Construction in the tangent plane of p_C for a stereo-camera as seen from above. Right: Examples of viewpoints and interpolated contour points constructed as formulated in Equation 1 and 2 with $k = L$ weighted with $\lambda_L \in [0.25, 0.5, 0.75]$ respectively.

4.2. Tangent Plane Construction

The following elaborates the concept illustrated in Figure 5, generalized for points p_k and viewpoints O_k with $k \in [C, L, R, U, D]$. Artificial points p'_k on the tangent plane produce equivalent projections q_k as points p_k when observed from O_k . Our interpolation concept exploits the linear correlation between the multiview window and the tangent plane: The same weights λ are used in Equation 1 to formulate an arbitrary view position as well as in Equation 2 to construct the corresponding artificial contour point.

$$O_A = O_C + \sum_k \lambda_k w_k \quad (1) \quad p'_A = p_C + \sum_k \lambda_k s'_k \quad (2)$$

$$\text{with } k \in [L, R, U, D]$$

Vectors $s'_k = p'_k - p_C$ are determined for each polyline point where p'_k is derived as formulated in Equation 3 for monoscopic-multiview. Points p_k^* from Equation 4 are used and explained in Section 4.3 for stereo-consistent results.

$$p'_k = p_k + \frac{(p_C - p_k) \cdot n_C}{v_k \cdot n_C} v_k \quad p_k^* = p_C + \frac{(p_k - p_C) \cdot n_k}{v_C \cdot n_k} v_C \quad (3) \quad (4)$$

In the top-view example from Figure 5, points p'_k could be specified as the intersection of the tangent lines at p_C and p_k . In object space, however, intersections have to be determined for the tangent line at p_k with the tangent plane at p_C given by $0 = (p' - p_C) \cdot n_C$. This is formulated in Equation 3 for point p'_k of a general contour point p_k on p_C 's tangent plane. So far, this concept simply extends the idea of intersecting tangent lines (Figure 5) to tangent planes in object space. In the top-view example, the vectors s'_L and s'_R span a line. In object space, the four vectors s'_L, s'_R, s'_U and s'_D span a planar polygon of four adjacent rhombi in the tangent plane of p_C .

4.3. Stereo Consistency

So far, our tangent plane concept is able to produce consistent contour interpolations in a monoscopic-multiview scenario. However, the created contours are not yet stereo-consistent which violates number 3 of the stated goals from Section 2.4. As one can recall from their definition: Points p'_L, p'_R, p'_U and p'_D are constructed to be

coplanar in p_C 's tangent plane. However, for stereo-consistency the points p'_L and p'_R should also be in the epipolar plane of O_L, O_R, O_C and p_C , which is not necessarily the case. A naive attempt to correct this issue could be, to project points p'_k into the epipolar plane where they should be. This would not only corrupt the initial geometrically correct concept but also introduce heavy artifacts like shape deformations and is therefore not a suitable option. Figure 5(d) (supplemental material) features an example with artifacts.

Geometric solution: The following approach is designed for correct epipolar geometry and is therefore able to establish stereo-consistency: In the top-view scenario of Figure 5, the new stereo-consistent concept is actually identical to our first one. In three-dimensional object space, however, things will be different. The tangent plane at p_C is not intersected with a tangent line of p_k but the other way round: The tangent line at p_C is used to intersect the tangent plane at p_k given by $0 = (p^* - p_k) \cdot n_k$. New points p_k^* are formulated in Equation 4. Points p_k^* and p_C are still coplanar but p'_L, p'_R, p_C and, respectively, p'_U, p'_D, p_C are now also collinear. Therefore, one can easily comprehend the alignment of points p_k^*, q_k, O_k ($k \in [L, C, R]$) in an epipolar plane.

In our general geometrically correct concept from Section 4.2, vectors s'_k span up a polygonal window of four rhombi around p_C . With this new approach, s'_L and s'_R are not necessarily linear independent from s'_U and s'_D . The example on the right in Figure 6 illustrates how the former window around p_C is now actually a line. Therefore, individual amplitudes of the s'_k vectors are now essential.

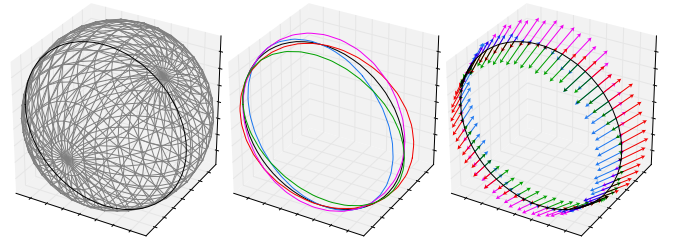


Figure 6: Wireframe of a sphere and center-view contour are shown on the left. Multiview contours as seen from O_L, O_R, O_U, O_D and O_C are shown in the middle. Corresponding tangent plane vectors s_k^* are shown on the right, lengths are scaled for the illustration only. Small variations in length are due to mesh tessellation.

Corresponding points are now by definition aligned in the same epipolar plane. Artifacts can arise in cases where the tangent plane at p_k and the tangent line at p_C are close to parallel. Thus, resulting intersections can be extremely far away. However, especially their extremeness makes it relatively easy to detect such outliers. From the observation of clean results, one can derive the assumption that the amplitude $|s_k^*|$ usually changes moderately from point to point along a contour polyline. Therefore, outlier amplitudes are corrected with a Median filter over a 1-ring neighborhood.

4.4. Arbitrary Camera Motion

The introduced tangent plane concept furthermore allows for simulating contours to move on objects as if they were observed from

an arbitrary camera path. This is achieved by concatenating simplified multiview windows and interpolating the camera path segment-wise. As illustrated in Figure 7 each key-camera is initialized with only one offset vector w_i pointing to the next key-camera. Contours are generated for these key-viewpoints and matched with the contours of their path-successor respectively. Furthermore, we switch from our skewed-frustum camera model back to a regular targeted perspective camera. Thus, we are able to sample this path with any desired resolution and can create a monoscopic but temporal-consistent line drawing animation. Glitches may occur at camera-handover points because topological contour differences are so far not covered in the interpolation. However, the transitions become smoother with more and closer cameras. E.g., 24 key-cameras are sufficient for the path of a semicircle around a simple and smooth object. On more advanced geometry, one could double the number of key-cameras to decrease artifacts. Animated examples are shown in the attached supplemental material.

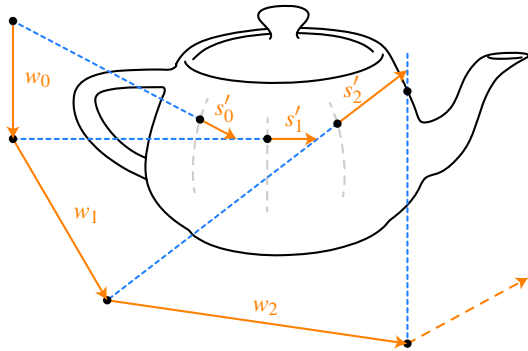


Figure 7: Construction of a fly-around camera path with key-cameras and according key-contours. As the camera path is sampled on w_i , the corresponding contours are interpolated with s'_i .

4.5. Mixed Contour Generators

As mentioned in Section 3, arbitrary contour generators can be used to generate polylines for our matching and alignment procedures. In an image space approach, the origin of contours is crucial information because different kinds of contours require to be treated separately. In our pipeline, however, various kinds of contours are mixed in the contour set of each view position respectively. For the procedures of matching and tangent plane interpolation, they are treated equivalently. View-independent contours match perfectly anyway, therefore vectors s'_k are just zero. Other view-dependent lines like suggestive contours are handled with the same assumption of surface smoothness as silhouette contours.

5. Visibility and Occlusion

As pointed out in many other publications [HZ00, RCDF08, BHK14], determining contour visibility is a complex task. Contours on triangle meshes occur on front-facing and back-facing triangles. Simple ray-tests will label points on back-facing geometry as not visible. Although this is geometrically true, local self-occlusions are not of interest at this point. A contour line should be visible as long as it is not occluded by any other surface area. Our solution to overcome the self-occlusion issue is to locally interpolate the objects geometry, not the contour polyline itself. A mesh

edge, featuring a contour point, is therefore approximated with a circular arc using cubic Bézier splines and the magic number determined by Goldapp [Go91]. This furthermore promotes the expectation of a smooth surface approximated by discrete geometry. The example in Figure 8 shows our achieved improvements regarding consistent visibility. Approaches equivalent to ours use Hermite splines [WTW*08, LWM15] instead of Béziers.

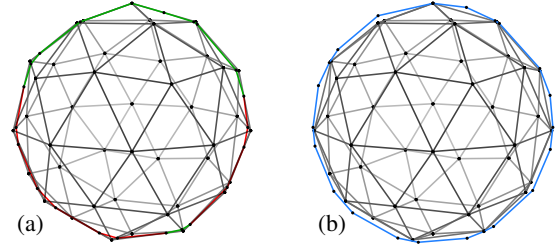


Figure 8: Silhouette contours on an icosphere. (a) Naive contour points with visible and invisible segments. (b) All points on Bézier interpolated contours passed the ray-test and are therefore visible.

Although Bézier interpolated geometry improves the self-occlusion issue in general, pointwise ray-tests can still feature false-positives and false-negatives. Therefore, we have to enforce the following general characteristics for the visibility of contour polylines:

1. Polylines can be completely, partly or not at all visible.
2. Visibility can only change at special points.

Hertzmann and Zorin [HZ00] categorize these special points as: Intersections of polylines in object space, respectively image space and at segments parallel to the view direction called cusp-points. We determine polyline intersections in their image space projection using a 2D kd-Tree. To eventually enforce the two rules stated above we employ a special data structure, the view graph. To avoid confusion with a visibility graph, the name view graph was adopted from Bénard et al. [BHK14] whereas in a planar case it is also called a view map [GTDS10].

View Graph: Contour polylines are translated into chains of nodes, each node aware of its own ray-test visibility state. 2D polyline intersections are inserted in the affected chains as linked special nodes, as illustrated in Figure 9. This link propagates visibility information between the chains. Special nodes partition a chain into individual sections. The visibility per section is determined via majority vote of its individual nodes (II in Figure 9). Visibility transitions around special points need to be treated separately for each adjoining section. Otherwise, gaps could occur between the last point of a chain and the special point. Chain sections can be foreground, background or occluded background. An iterative application of our set of rules (supplemental material) resolves visibility for special nodes and associated chain sections. Remaining ambiguous junctions are resolved as the final step (III in Figure 9).

5.1. Center Dominant Visibility (CDV)

The visibility and view graph procedures were introduced considering only one view point. We can exploit some characteristics of those algorithms to extend their capability for multiview: Contours are interpolated for an intended viewpoint as described in Section 4.1. View-dependent special points are determined for this

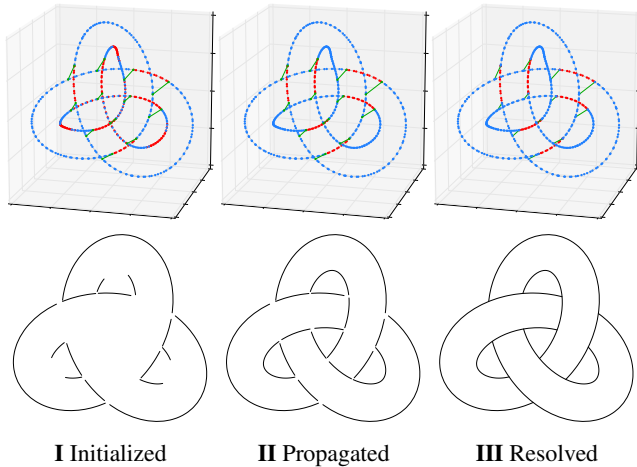


Figure 9: Three states of a view graph during its lifetime and their according projections. Legend: **visible**, **invisible**, **special node**.

viewpoint and altogether used to initialize the view graph. Visibility is obviously also view-dependent, but not necessarily required to be sampled separately: Ray-tests are performed only once for the center view contour and the results can be re-used for all other multiview positions. As illustrated in the top-half of Figure 10, re-used visibility samples from the center do not match with the aligned contours from left and right view. However, since the view graph’s designated task is to establish section-wise uniform visibility, it is also capable to resolve these issues. As shown in the bottom-half of Figure 10, false-positives and false-negatives are resolved: Correct view-dependent occlusions are established, even for arbitrary view positions without re-sampling visibility.

6. Discussion

In this section we discuss the performance of our proposed pipeline. Advantages of our novel object space concept over established image space approaches are elaborated in Section 6.2. Our proposed framework covers many issues like hidden lines or mixed contour generators which could so far not be handled properly.

6.1. Results

Figure 11 shows the outcome of a naive approach compared to our stereo-consistent results. The line drawings on the left in Figure 11 were generated with the same setup as the anaglyphs on the right, only with more disparity. Due to the wider offset between left and right camera, shape dissimilarities of contour lines are now much more prominent as denoted with the red circles. The green circles on our results point out the same contour lines which are now coherent in shape, regardless of the wider disparity. Unnaturally wide disparity can be unpleasant, so it was chosen to be small for the anaglyph examples on the right. Since the left and right camera were not so far apart, the differences in shape between left and right contours are not tremendous. However, there are contours in the individual-image that disturb the depth impression with binocular rivalry. These lines occur exclusively in the left or right image and therefore violate condition 1 and 2 from Section 2.4. Our stereo-consistent result does not contain such rivalries.

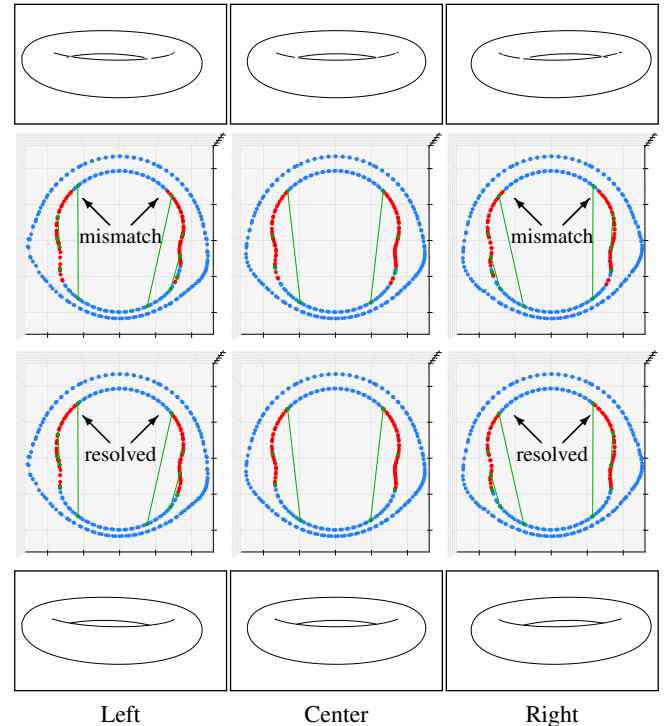


Figure 10: View graph plots and resulting projections. Top: Initialization with visibility from the center view might result in mismatches. Bottom: Final states with correct resolved visibility. Legend: **visible**, **invisible**, **special node**.

A direct comparison of our results to image space solutions of Kim et al. [KLKL13] is featured in Figure 12. Due to unknown parameters for suggestive contours, our results feature a little more details but in terms of stereo-consistency both results are of equal quality.

6.2. Object vs. Image Space

Compared to established image space solutions for stereo-consistency, our object space approach with generalized multiview capabilities yields many advantages as described in the following.

Hidden Lines: The approach by Kim et al. for *Stereoscopic 3D Line Drawing* [KLKL13] is also center-eye based but operates solely in image space. Challenges that arise in the image space are natively covered with our proposed object space approach. In their concept, special treatment is required for *hidden lines*, which are visible for one eye but hidden for the other. Kim et al. have to remove hidden lines for some steps of their algorithm and deal with them later, how is not further elaborated. In contrast, our solution features hidden lines without further ado because visibility and occlusion are resolved individually for each final view with the concept introduced in Section 5.1. Examples of hidden lines in our results are pointed out in Figure 13. Figure 14 directly compares our results to Kim et al. [KLKL13].

Mixed Contour Generators: Once a contour line is projected and rendered as an image, it is unclear if it was a view-dependent or view-independent contour. Kim et al. noted that they have to handle these cases separately from each other. In our object space

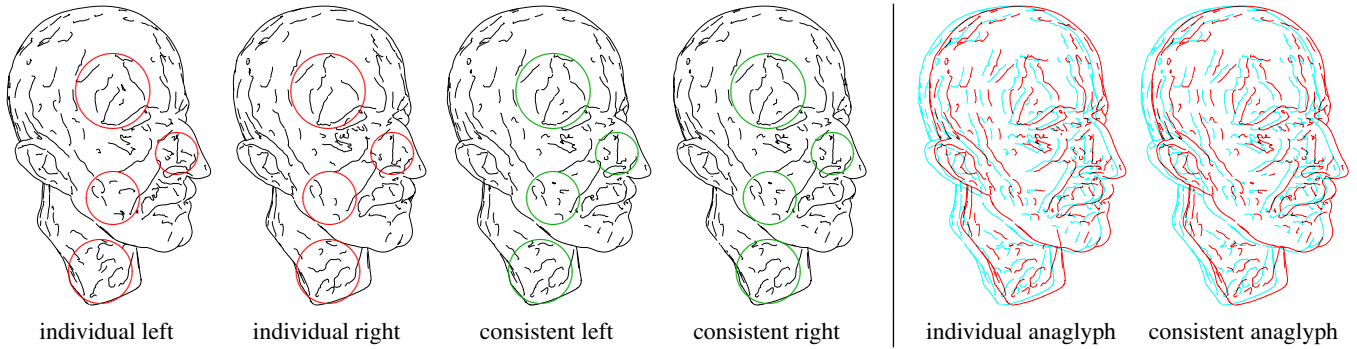


Figure 11: Silhouette contours and suggestive contours on the *Max Planck Bust* [DFRS03a]. Left and right views of an inconsistent approach along with our consistent results are shown on the left and anaglyph examples on the right. Results on the left are created with more disparity to exaggerate the denoted inconsistencies, examples on the right feature less disparity for a more pleasant anaglyph experience.

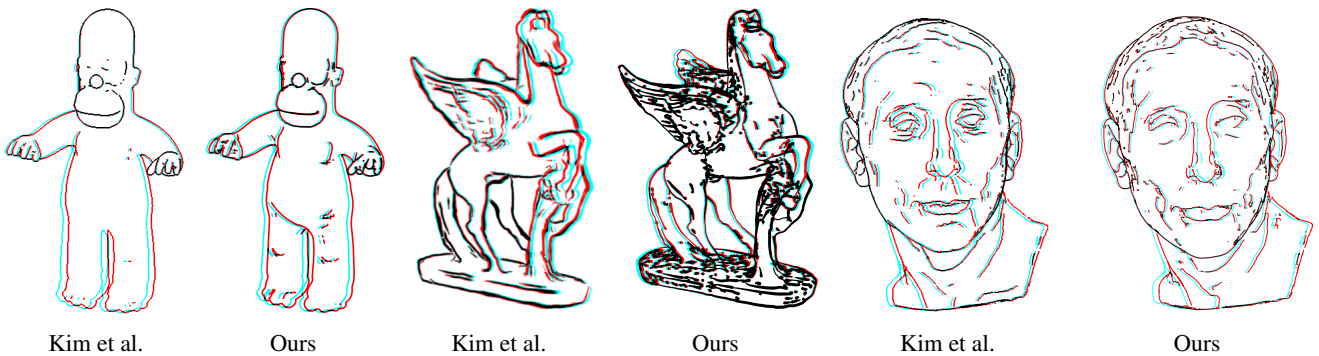


Figure 12: Result comparison to Kim et al. [KLLK13]. 3D models were provided, camera parameters were approximated manually.

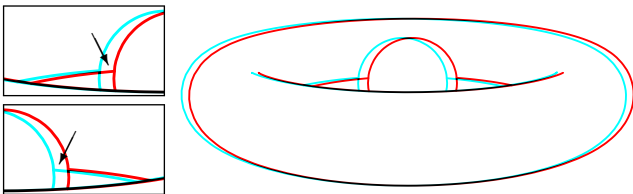


Figure 13: Boxes on the left point out *hidden lines*. Our CDV view graph concept properly resolves view-dependent occlusions.

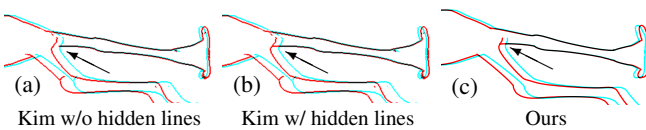


Figure 14: Comparison of results of Kim et al. [KLLK13] with hidden line removal (a), without hidden line removal (b) and our result (c) with natively incorporated hidden lines.

framework, however, both types of contours are treated equivalently. Therefore, they can be used together in the same scene as shown in Figure 15. View-independent lines will obviously find perfect matches for all view positions. Tangent-plane offset vectors (s_k^* from Section 4.1) are determined for all contour points nevertheless. For view-independent contours these vectors are just zero.

Object-based Stylization: In the image space solution of Kim et al. [KLLK13], stylization is realized by propagating style parameters from left to right. However, in *Coherent stylized silhouettes* [KDMF03] the stylization is based on properties of an object’s 3D shape. The relationship between projected line in the result image and its original 3D polyline is embodied in our view graph concept. Therefore, characteristics like curvature, z-depth or point-density can be extracted in object space to be used for stylization of the resulting image space projections. In Figure 1 (right) and Figure 12 (center, Ours), relative z-depth scaled the stroke-width.

Large-scale Camera Motion: Kim et al. denoted temporal consistency as the next step for line drawings, which cannot be achieved in image space. Section 4.4 elaborates on how to interpolate contours as observed from an arbitrary camera path with our multiview window. Results are shown in our supplemental material.

6.3. Performance and Limitations

Realtime: The pipeline breakdown from Table 1 is exemplary for the generation of two result images (L, R). Curvature analysis was precomputed, loaded from files and excluded from the list. As one can observe, the most time-consuming steps are the generation of view-dependent contours and visibility related operations. However, contours are generated only for a fixed number of view positions: O_L, O_R and O_C for a stereo setup, O_L, O_R, O_U, O_D and O_C for the whole multiview window. After contour generation and matching,

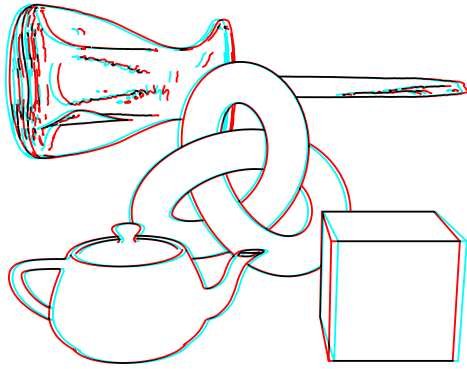



Figure 15:  View-independent creases as well as view-dependent silhouette and suggestive contours mixed in the same scene. Object Sources: Screwdriver [INR04], Utah Teapot [Kno14]

sampling a position from the multiview window only requires to align contours and to project them accordingly. These relatively simple vector operations can be computed very efficiently. For large meshes as the *Max Planck Bust* (98260 faces) or the *Stanford Bunny* (69630 faces) this can be done multiple times per second on CPU. For meshes of lower resolution (<20000 faces) our implementation was able to reach real-time performance with more than 24fps. This actually allows us to move within the multiview window and simultaneously create a temporally coherent stream of view-dependent contours. However, this real-time experience can only be achieved as long as there is no view graph involved. To establish correct view-dependent occlusions, the view graph algorithms alone take about 0.25s for the contours of a smaller mesh like the *Utah Teapot* (2464 faces) and about 5s for a large mesh as the *Stanford Bunny*.

Task	Time
Loading and Initialization	4.22%
Contour generation (O_L, O_C, O_R)	61.09%
Visibility Ray-Tests (center only)	12.32%
Contour Matching ($C_L \rightarrow C_C, C_R \rightarrow C_C$)	2.93%
Contour Alignment (L, R)	1.30%
View Graph Initialization (L, R)	3.44%
Image Space Intersections (L, R)	14.37%
Visibility Propagation (L, R)	0.32%

Table 1: Consumed time for individual pipeline steps, exemplary for a stereo-setup with two result views. Measured with 10 objects of different face-count using silhouette and suggestive contours.

Matching issues: Robust pointwise matches between contour lines are crucial for the success of the tangent plane concept from Section 4.1. Pointwise nearest neighbor search was finally used in all shown examples. It has low overhead and is able to produce satisfying results in many scenarios but also has its limits. However, more severe artifacts may occur on under-tessellated meshes where relative positions of the same contour line can be quite far apart for different viewpoints. Nevertheless, these contours are matched to the *closest* points available in the other contour. But in some cases, *closest* is not sufficient. This can cause projections on a wrong tangent plane to float around heavily, as the view position moves in the

multiview window. Exemplary failure cases are illustrated in Figure 16 with color-emphasized eyebrows. Such artifacts can usually be avoided with a smaller multiview window or a mesh of higher resolution (see supplemental material).

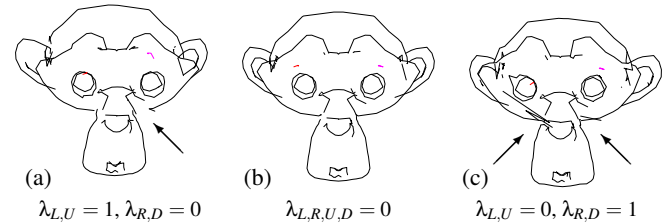


Figure 16: Examples from different multiview-window positions of matching artifacts due to an under-tessellated mesh and too wide disparity. Complete figure is featured in the supplemental material.

View graph glitches: The view graph propagates visibility along chains based on the information of individual nodes. Especially the algorithms to resolve ambiguous states on junctions depend on information from neighboring nodes. Glitches illustrated in Figure 17 can occur due to insufficient data. Low mesh resolution in 17a caused contour points to be far apart. With more faces in 17b, junctions were resolved properly because of sufficient locally relevant neighbors. Image space intersections in 17c happen to form an inconvenient constellation and are not resolved properly. In 17d, the same cubes are closer to each other and corner became resolvable.

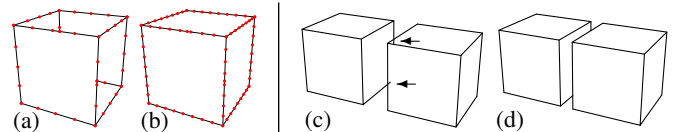


Figure 17: View graph glitches: Occlusions are falsely resolved for (a) but are correct for (b). Cubes in (c) and (d) are identical, the distance between them is different. Artifacts are pointed out in (c).

6.4. Outlook

Matching contour points with nearest-neighbors works sufficiently well in most scenarios but also has its limits. On rough meshes or with a too wide multiview window, corresponding contours may be too far apart and wrong points are matched. More advanced matching algorithms for 3D polylines could prevent such artifacts even for extreme scenarios. Topological differences between contours can be interpolated but are so far not implemented. This challenge could be faced with a proper adaptation of the vector animation complex [DRvdP15]. A faster and more general solution to the visibility problem for line drawings would be a great advance for many applications. For our framework it would decrease visibility artifacts and promote interactive usage of multiview line drawings.

7. Conclusion

Our introduced tangent plane concept seamlessly interpolates view-dependent contours between different viewpoints for stated assumptions of surface smoothness. This covers our stated goal of

flexible view positions even for view-dependent contour generators. Stereo-consistency is achieved with a modification of our first approach but comes with a trade-off. Interpolated contours for a view position in the multiview window are stereo-consistent in terms of epipolar geometry but their alignment on surface tangents is no longer guaranteed to be completely accurate. Nevertheless, compared to outcomes of a naive concept, the results of our approach are a clear improvement. Achieved stereo-consistency is also comparable to competitive publications [KLKL13] in the field of line drawings. However, our fully vectorized and multiview-generalized object space pipeline yields many advantages over established image space approaches, such as: Stereo-consistent multiview and fly-around camera path, native support of hidden lines, correct view dependent occlusions, mixed contour generators and stylization based on object space features.

References

- [BHK14] BÉNARD P., HERTZMANN A., KASS M.: Computing smooth surface contours with accurate topology. *ACM Transactions on Graphics (TOG)* 33, 2 (2014), 19. 2, 6
- [CGL*08] COLE F., GOLOVINSKIY A., LIMPAEACHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? In *ACM Transactions on Graphics (TOG)* (2008), vol. 27(3), ACM, p. 88. 2, 3
- [DFRS03a] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive Contour Gallery, 2003. Online; accessed August-2016, <http://gfx.cs.princeton.edu/proj/sugcon/models/>. 2, 8
- [DFRS03b] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 848–855. 2
- [DRvdP15] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector graphics animation with time-varying topology. *ACM Trans. Graph.* 34, 4 (2015). 9
- [GD03] GUIGUE P., DEVILLERS O.: Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of graphics tools* 8, 1 (2003), 25–32. 2
- [GGSC98] GOOCH A., GOOCH B., SHIRLEY P., COHEN E.: A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 447–452. 2
- [Gol91] GOLDAPP M.: Approximation of circular arcs by cubic polynomials. *Computer Aided Geometric Design* 8, 3 (1991), 227–238. 6
- [GTDS10] GRABLI S., TURQUIN E., DURAND F., SILLION F. X.: Programmable rendering of line drawing from 3d scenes. *ACM Transactions on Graphics (TOG)* 29, 2 (2010), 18. 6
- [Her98] HERTZMANN A.: Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 453–460. 2
- [HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 517–526. 2, 6
- [INR04] INRIA: Models from Visionair Shape Repository, 2004. Online; accessed August-2016, <http://visionair.ge.imati.cnr.it/ontologies/shapes/view.jsp?id=16-Casting>, <http://visionair.ge.imati.cnr.it/ontologies/shapes/view.jsp?id=40-Screwdriver>. 9
- [JDA07] JUDD T., DURAND F., ADELSON E. H.: Apparent ridges for line drawing. *ACM Trans. Graph.* 26, 3 (2007), 19. 2
- [KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22(3), ACM, pp. 856–861. 8
- [KH11] KARSCH K., HART J. C.: Snaxels on a plane. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (2011), ACM, pp. 35–42. 4
- [KLKL13] KIM Y., LEE Y., KANG H., LEE S.: Stereoscopic 3d line drawing. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 57. 1, 2, 7, 8, 10
- [Kno14] KNOWLES P.: Common 3D Models/Meshes use in Computer Graphics Research, 2014. Online; accessed August-2016, <http://goanna.cs.rmit.edu.au/~pknowles/models.html>. 9
- [Koo09] KOOIMA R.: Generalized perspective projection. *LSU Computer Science and Engineering Division*. <http://csc.lsu.edu/~kooima/articles/genperspective/June> (2009). 2, 3
- [KW78] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *International journal of computer vision* 1, 4 (1988), 321–331. 4
- [LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 31–42. 3
- [LMY*13] LIU X., MAO X., YANG X., ZHANG L., WONG T.-T.: Stereoscopizing cel animations. *ACM Transactions on Graphics (SIGGRAPH Asia 2013 issue)* 32, 6 (November 2013), 223:1–223:10. 2
- [LWM15] LOU L., WANG L., MENG X.: Stylized strokes for coherent line drawings. *Computational Visual Media* 1, 1 (2015), 79–89. 4, 6
- [Mar00] MARKOSIAN L.: *Art-based modeling and rendering for computer graphics*. Brown University, 2000. 2
- [Mö197] MÖLLER T.: A fast triangle-triangle intersection test. *Journal of graphics tools* 2, 2 (1997), 25–30. 2
- [NAK12] NORTHAM L., ASENTE P., KAPLAN C. S.: Consistent stylization and painterly rendering of stereoscopic 3D images. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (2012), pp. 47–56. 2
- [NAK13] NORTHAM L., ASENTE P., KAPLAN C. S.: Stereoscopic 3d image stylization. *Computers & Graphics* 37, 5 (2013), 389–402. 2
- [NM00] NORTHRUP J., MARKOSIAN L.: Artistic silhouettes: a hybrid approach. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (2000), ACM, pp. 31–37. 2
- [OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM transactions on graphics (TOG)* 23, 3 (2004), 609–612. 2
- [Rad99] RADEMACHER P.: View-dependent geometry. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 439–446. 2
- [RCDF08] RUSINKIEWICZ S., COLE F., DECARLO D., FINKELSTEIN A.: Line drawings from 3d models. In *ACM SIGGRAPH 2008 classes* (2008), ACM, p. 39. 2, 3, 6
- [RŠDD11] RICHARDT C., ŚWIRSKI L., DAVIES I. P., DODGSON N. A.: Predicting stereoscopic viewing comfort using a coherence-based computational model. In *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging* (2011), ACM, pp. 97–104. 2
- [SG05] STAVRAKIS E., GELAUTZ M.: Stereoscopic painting with varying levels of detail. In *Electronic Imaging 2005* (2005), International Society for Optics and Photonics, pp. 450–459. 2
- [WTW*08] WANG L., TU C., WANG W., MENG X., CHAN B., YAN D.: Silhouette smoothing for real-time rendering of mesh surfaces. *IEEE transactions on visualization and computer graphics* 14, 3 (2008). 6
- [YBS05] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: Fast and robust detection of crest lines on meshes. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling* (2005), ACM, pp. 227–232. 2