Local Surface Approximation Contours for Virtual Reality Stylisation

Amir Zaidi, Ricardo Marroquim, Michael Weinmann, and Elmar Eisemann

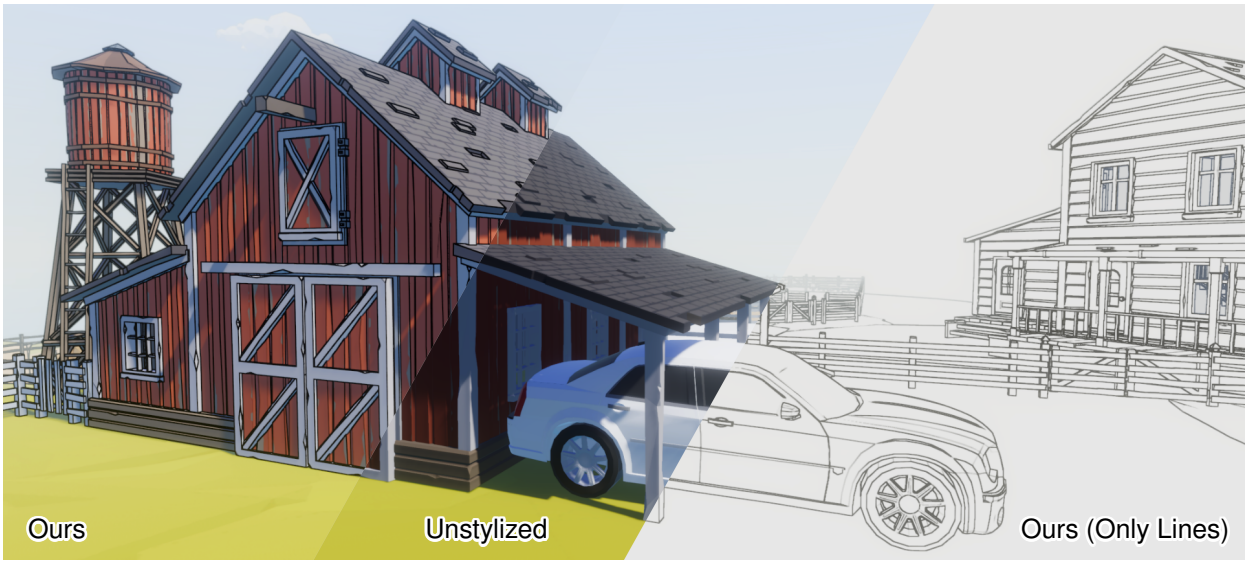

Fig. 1: LSA contours automate stereo-aware VR stylisation on arbitrarily complex scenes in constant time.

Abstract— Line art is an illustrative technique with a wide use in education and art. In the context of image abstraction, its potential for increasing memorisation and recognition has been demonstrated, which motivates its use in scientific illustrations. While much work has focused on the conversion of 3D models into a line-art representation, there is a lack of solutions for virtual reality. Applying existing methods for each eye independently turns out to fall short due to cost constraints, distracting artifacts due to inconsistencies, or limitations regarding the input geometry.

To address these limitations, we present a contour renderer for virtual reality. It operates in screen space, making it flexible, yet it relies on a local surface approximation combined with a registration error metric for robustness. Inconsistent occluding contours are continuously merged, and lines with no correspondence between both eyes are culled. The method is easy to implement, highly efficient even for high-resolution imagery, and, according to user evaluations, avoids the noticeable artifacts produced by existing work.

Index Terms—Contours, stylisation, virtual reality, stereoscopy

1 Introduction

Lines play an important role in rendering stylisations, using curves as the primary method to capture and accentuate object shapes (Fig. 1). Such stylisations can simplify the representation of an object and, for 2D illustrations, it has even been shown that the use of such an abstract representation can be beneficial for memorisation and recognition [32]. Yet, it has not yet been explored whether the same holds for 3D rendering algorithms. This is of particular interest as we rely increasingly on rendering, e.g., via virtual and augmented reality (VR/AR) for educational purposes. Yet, even educational VR/AR often employs realistic shading models. One reason is that rendering clean contours in a stereoscopic setting is challenging. Applying existing contour-rendering methods for each eye independently can lead to distracting artifacts, involve high computational costs, or impose limitations with respect to input geometry [3].

We propose **Local Surface Approximation Contours** (LSA contours), a high-quality and efficient approach to stereo-compatible contour-like feature lines, which avoids the inconsistencies of prior methods. Further, we illustrate the benefit of this technique for VR via a user study, confirming the potential for increased memorisation and recognition when relying on abstract rendering.

Specifically, we make the following contributions. First, we present an efficient and robust screen-space solution for contour-like feature lines. Second, we adapt this approach to stereo, preventing binocular rivalry and staying temporally coherent as well as spatially consistent. Third, we compare to the state-of-the-art and illustrate potential benefits of stereo contours in a user study. We will release both source code and a demo program on acceptance.

2 Related Work

Line art has received much attention in non-photorealistic rendering. Stylisation systems commonly rely on extracted feature lines [10, 13]. In the following, we detail key methods for feature line extraction, and refer the reader to Bénard and Hertzmann [6] for an extensive overview.

The earliest approaches relied on vector graphics as a line-art representation. Line art based on contours was considered a useful means for illustration. Besides robust object-space solutions [2], more efficient screen-space methods [22] typically operate on image buffers. In our work, we fuse robustness and efficiency in a screen-space method. Prior work merges depth and normal G-buffer channels [15, 27] but

- *Amir Zaidi is with Delft University of Technology. E-mail: a.zaidi@tudelft.nl.*
- *Ricardo Marroquim is with Delft University of Technology. E-mail: r.marroquim@tudelft.nl.*
- *Michael Weinmann is with Delft University of Technology. E-mail: m.weinmann@tudelft.nl.*
- *Elmar Eisemann is with Delft University of Technology. E-mail: e.eisemann@tudelft.nl.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

suffers from inconsistency in feature detection, requiring adjustments for different scenes and objects.

Besides contours, where the normal is orthogonal to the viewing direction, other feature lines exist. Examples include ridges and valleys [20], defined as minimum and maximum principal curvature lines, which our solution also supports. Other definitions include suggestive contours [9] and apparent ridges [16]. The former uses zero-crossings of radial curvature. The latter uses a view-dependent curvature definition based on the projected normal variation. These curves are shown to improve shape perception [7]. While not part of our contribution, we augment our feature-line detection with additional shading cues following recent findings [31, 32].

Performance and the spatio-temporal coherence of extracted lines are important aspects for VR. Screen-space approaches struggle with occlusion and aliasing due to rasterization artifacts; view-dependent feature lines are thus often temporally inconsistent. For stereo-consistency, object-space solutions exist [8], but these do not transfer to screen-space approaches. Even simple contour lines are not stereo-consistent [3]; a cube might be seen solely from the front in one eye and also from the side in the other, leading to discomfort.

Problems with stereo-consistency likewise occur with SSAO [24], where reprojection [21, 33] is used to compare views. Besides reprojection, an alternative is to remove the view-dependency from the feature lines [4, 5, 14, 18]. Often this involves a single view from where feature lines are reprojected, or several views contributing to a canonical curve. Such solutions either involve costly geometric computations or rely on screen-space solutions that often lead to artifacts due to occlusions and imprecision. A hybrid of screen- and object-space solutions provides geometric interpretation to the depth buffer, by fitting a plane to a local region of the buffer [26, 27]. Our method employs this interpretation, operating in screen-space but ensuring consistency in a robust manner.

We finally remark on the compatibility of line extraction with methods such as bump, normal, or parallax mapping [11, 17, 29]. These texture-based methods give the illusion of geometric surface details, which should affect feature lines. This is not the case for all solutions, as object-space methods often derive lines exclusively from the triangle mesh. Methods which merge G-buffers can implicitly integrate such information, as can our method.

3 Methodology

We propose a screen-space approach using a local surface approximation (LSA) paired with a registration error. Our method determines which pixel might contain a feature line, based on when the pixels' underlying geometry cannot be registered to such an LSA.

For a given pixel position p, we denote the underlying 3D world-space position and 3D surface normal as P_p and N_p respectively. As feature lines are local and analyzing a large region around p increases computational costs, we only consider the direct 8-pixel neighborhood window $\mathrm{W}(p)$ around p. In this neighborhood, we try to locally approximate the sampled surface.Yet, instead of a single approximation, we opt for an LSA definition between p and each pixel $q \in \mathrm{W}(p)$. Hereby, the number of LSA parameters is kept low while we retain high expressiveness. Given a LSA $\mathrm{S}(p,q)$ between p and q, we denote the registration error as:

$$\mathrm{E}(p) = \max_{q \in \mathrm{W}(p)} e(q, \mathrm{S}(p,q)), \tag{1}$$

where e measures deviation error of P_q and N_q to $\mathrm{S}(p,q)$. If all neighbours are predicted by their LSA, $E(p) = 0$. As one neighbour deviates, $E(p)$ increases, indicating the likely presence of a line.

In practice, we apply $\mathrm{E}(p)$ as a mask to modulate the rendering by inverting and multiplying with the rendered image. We can add some flexibility to $\mathrm{E}(p)$ in order to control the amount of visible feature lines by remapping, but concrete choices depend on the definition of $S(p,q)$ and e, which we describe in the following.

3.1 Sphere Fitting as LSA

Various choices for $S(p,q)$ present themselves (e.g. polynomial, spline, general quadratic). We instead opt for a fitted sphere, given its computational simplicity and link to curvature. We fit this sphere on the 3D positions and normals of pixels p and $\check{q}$, where $\check{q}$ is the pixel opposite to q in $\mathrm{W}(p)$. Note that we do not fit the sphere directly to q. Instead, we generate a sphere to test q using $\check{q}$.

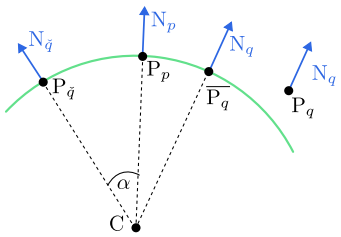

Fig. 2: The LSA $S(p,q)$ (in green) is tangent to $\mathrm{P}_{\check{q}}$ and P_p. The deviation error is proportional to the distance between P_q and the expected position $\overline{\mathrm{P}_q}$.

A sphere (C,r) is uniquely defined by its centre $C \in \mathbb{R}^3$ and radius r (four parameters). As an LSA forms a piece-wise reconstruction in a small area, we use negative r to represent negative curvature of the surface. We restrict the centre C to lie on the line through P_p travelling in direction N_p:

$$C = \mathrm{P}_p - r\mathrm{N}_p \tag{2}$$

We could simply constrain the sphere such that P_p and $\mathrm{P}_{\check{q}}$ lie on it, but note that normal $\mathrm{N}_{\check{q}}$ provides additional surface information; if the underlying surface is spherical, a sphere tangent to $\mathrm{P}_{\check{q}}$ and P_p would share the normals at these locations (Fig. 2). It follows that

$$\begin{aligned} |r| &= \frac{|\mathrm{P}_{\check{q}} - \mathrm{P}_p|}{2\sin(\frac{1}{2}\angle(\mathrm{N}_p, \mathrm{N}_{\check{q}}))} \\ &= \frac{|\mathrm{P}_{\check{q}} - \mathrm{P}_p|}{|\mathrm{N}_{\check{q}} - \mathrm{N}_p|}, \end{aligned} \tag{3}$$

thus r is the radius of the tangent sphere, given this spherical surface. Its sign corresponds to the sign of the surface curvature, which is positive in convex and negative in concave areas, i.e.

$$r = |r|\,\mathrm{sgn}((\mathrm{N}_{\check{q}} - \mathrm{N}_p) \cdot (\mathrm{P}_{\check{q}} - \mathrm{P}_p)),$$

where $\mathrm{sgn}(x)$ is the Signum function. The LSA is then uniquely defined by (C,r). As $r = \infty$ on planar surfaces, we compute curvature $\kappa = \frac{1}{r}$ in practice. The result is finite, and $\kappa = 0$ on planar surfaces.

3.2 Deviation Error Function

Given curvature, we define our first deviation error variant e_{curv} as:

$$e_{\mathrm{curv}}(q, \mathrm{S}(p,q)) := \kappa |\overline{\mathrm{P}_q} - \mathrm{P}_q|, \tag{4}$$

where normalization by κ makes e scale-invariant, and $\overline{\mathrm{P}_q}$ is the expected position on a spherical surface given correct N_q. If $\overline{\mathrm{P}_q}$ is on the sphere and N_q is the expected normal (Fig. 2), $\overline{\mathrm{P}_q}$ can be found by following the normal direction from the sphere centre, i.e. $\overline{\mathrm{P}_q} = C + r\mathrm{N}_q$. This leads to:

$$e_{\mathrm{curv}}(q, \mathrm{S}(p,q)) = |\kappa(\mathrm{P}_q - \mathrm{P}_p) - (\mathrm{N}_q - \mathrm{N}_p)| \tag{5}$$

Finally, we linearly remap and clamp $\mathrm{E}(p)$ from $[\eta_{\min}, \eta_{\max}]$ to $[0,1]$. We empirically select $\eta_{\min} = \frac{1}{4}, \eta_{\max} = \frac{1}{2}$.

This leads to a high-quality feature-line detector on curved surfaces and is efficient to compute. However, we note the following. Planar surfaces currently lead to missing feature lines, while fine mesh details result in aliasing. Further, this approach does not ensure stereoscopic consistency. We next address these points.

4 Extensions

We now extend our method to address (1) monoscopic and (2) stereoscopic artifacts. We tackle each category separately.

Fig. 3: e_{curv} does not detect parallel planes. e_{plane} explicitly searches for these, and adds missing lines to the final result.

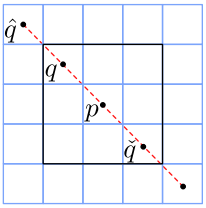

Fig. 4: Position of $\hat{q}$ if q is the top-left (NW) neighbour of p. The neighbourhood of p is illustrated with the black outline. The dotted red line represents the cardinal direction SE-NW and crosses the 5 pixels in this direction used for the Laplacian function.

4.1 Monoscopic Improvements

Planarity The deviation error e_{curv} detects discontinuities on curved surfaces. Yet, a discontinuity between two parallel planes ($\mathrm{N}_q = \mathrm{N}_p$) results in $e_{\text{curv}}(q, \mathrm{S}(p,q)) = 0$ regardless of $\mathrm{P}_q - \mathrm{P}_p$, leading to a missed feature line. To detect strong shifts in position along the normal direction, we define a new error function e_{plane} as:

$$e_{\text{plane}}(q, \mathrm{S}(p,q)) := k\left(\frac{\min_{\mathrm{N}_- \in \text{arc}(\mathrm{N}_p, \mathrm{N}_q)} |\mathrm{N}_- \cdot (\mathrm{P}_q - \mathrm{P}_p)|}{\max_{\mathrm{N}_+ \in \text{arc}(\mathrm{N}_p, \mathrm{N}_{\check{q}})} |\mathrm{N}_+ \cdot (\mathrm{P}_{\check{q}} - \mathrm{P}_p)| + \varepsilon}\right), \quad (6)$$

where $\text{arc}(X,Y)$ is the set of all spherical interpolated normals between normals X and Y and $\varepsilon = 0.00001$. k is a map, linearly transforming and then clamping $[50, 150]$ to $[0,1]$. We selected these constants empirically to deliver a good detection of planar shifts. If two neighboring pixels reside on the same flat plane, the denominator simplifies to ε, which means that e_{plane} grows rapidly if another pixel is not on the same plane.

We can combine terms into a merged error function (Eq. (1)) as:

$$e_{\text{sum}}(q, \mathrm{S}(p,q)) := e_{\text{curv}}(q, \mathrm{S}(p,q)) + e_{\text{plane}}(q, \mathrm{S}(p,q)), \quad (7)$$

Geometric Aliasing Another issue is aliasing, which can cause rapid flickering and stems from small geometric features. For example, a subpixel gap between polygons may or may not be captured in the image buffer, potentially resulting in missing discontinuities. As a heuristic remedy, we involve pixel $\hat{q}$, the pixel twice as far from p as q in the same direction $q - p$ (Fig. 4). The idea is to test and construct LSAs with these more distant pixels to see if the LSA error remains consistent. Specifically, we define $a := e_{\text{sum}}(\hat{q}, \mathrm{S}(p,q))$, $b := e_{\text{sum}}(q, \mathrm{S}(p,\hat{q}))$ and $c := e_{\text{sum}}(q, \mathrm{S}(p,q))$ and remap and clamp the three terms, just like e_{curv}. We then define a new error function:

$$e_{\text{aa}}(q, \mathrm{S}(p,q)) := \begin{cases} (\min(a,b) + c)/2 & \text{if } min(a,b) < c \\ c & \text{otherwise.} \end{cases} \quad (8)$$

Acceleration Instead of operating on an eight pixel neighborhood for subsequent operations, we can consider half and merge the other half using

$$e_{\max}(q, \mathrm{S}(p,q)) := \max(e_{\text{aa}}(q, \mathrm{S}(p,q)), e_{\text{aa}}(\check{q}, \mathrm{S}(p,\check{q}))).$$

In this way, we effectively halve our computational cost.

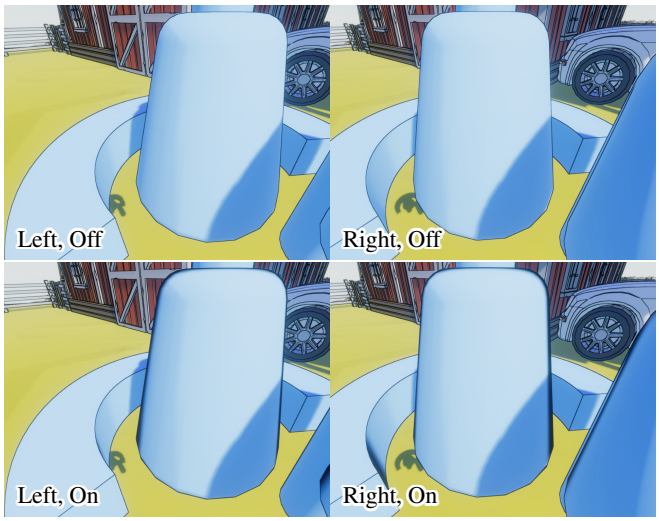

Fig. 5: Illustration of reducing binocular rivalry with a contour fade-out. Note the soft fade on the sides of the objects.

Line Density When moving away from a geometrically complex object, the density of its constructed feature lines increases. This is expected, but visually unpleasing. Offline methods for line art explicitly address this [12], and we apply filtering to do the same. For all 8 directions around a pixel, we use a local-maximum filter, reducing $e_{\max}$ by twice the smallest value of $e_{\max}$ in a line of 5 pixels centered at p (Fig. 4).

4.2 Stereoscopic Compatibility

The resulting line art from the monoscopic method is temporally coherent: the detected features are robust, providing smooth transitions during camera movement. However, VR necessitates coherency across views, which is not yet the case. This is crucial for depth perception and to avoid binocular rivalry.

We identify two main causes for the latter; view-dependent occluding contours, and mismatched rasterization artifacts (aliasing) causing mismatched feature detection. To address the former, we introduce a contour fade-out gradient per view. To address the latter, we suppress lines that we identify as inconsistent using reprojection.

Contour Fade-Out A viewer experiences binocular rivalry when there are significant differences between left and right views. We can facilitate binocular fusion of line art by adding stereo-consistent low-frequency information, e.g. diffuse shading. This helps the visual system identify a phase to map between viewpoints. However, diffuse shading appearance depends on lighting conditions, and can clash with the visual style created by line art.

We selectively add a smooth transition near contours, similar to diffuse shading from a light source located at the viewer (Fig. 5). To determine possible contours for the viewpoints of the users, we do not directly use $\mathrm{V_L}$ and $\mathrm{V_R}$, the position of the left and right view origins. The reason is that if the user rotates their head around the camera forward vector F, the contour fade-out should stay consistent. Thus, we instead consider all viewpoints in a disc A on the plane orthogonal

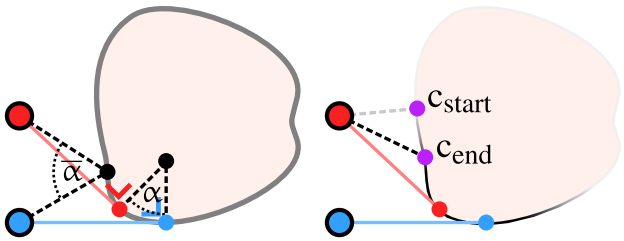

Fig. 6: Left: Estimating the angular difference between contours (α) from the angle between eyes at arbitrary surface points ($\overline{\alpha}$). These are approximately equal for objects significantly smaller than the distance to the eyes. The outlined red and blue circles indicate the left and right eyes. Right: $\overline{\alpha}$ determines c_{start} and c_{end} to control the contour fade-out spread (exaggerated here for visual clarity).

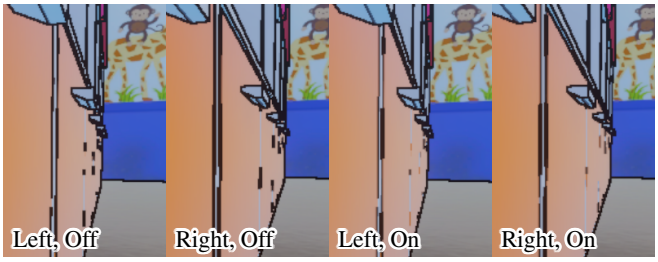

Fig. 7: Suppression of stereo-incoherent noise using reprojection and aliasing reduction. TAA is disabled to accentuate the effect.

to the view direction, whose center is the middle point $\frac{V_L+V_R}{2}$ and its radius $|\frac{V_L-V_R}{2}|$. A naturally includes V_L and V_R on its boundary.

For every pixel p, we define $V_{C,p}$ as the view in A closest to seeing the surface in p as a contour (minimizing the cosine between view direction and normal). This view usually lies on the boundary of A. We then define $\phi(p) = \angle(V_{C,p} - P_p, N_p)$, the angle between the surface normal and the ray from the view $V_{C,p}$ towards P_p.

We first determine the disocclusion due to the discontinuity. When $\phi(p) \geq \frac{\pi}{2}$, the shading must be dark to fill in the region between the contours of the left and right eye. When $\phi(p) < \frac{\pi}{2}$, the dark shading should rapidly become transparent. This is achieved with a linear mapping from $[\frac{\pi}{2} - c_{start}, \frac{\pi}{2} - c_{end}]$ to $[0,1]$, where c_{start} and c_{end} determine at which angle the shading gradient should be invisible and at which angle the gradient should be at maximum intensity respectively. To define the actual gradient of the fade-out, we find $\overline{\alpha(p)} = \angle(V_{L,p} - P_p, V_{R,p} - P_p)$, as an approximation for the angle between the surface patches containing the contour lines for the two views $\alpha(p)$ (Fig. 6). Specifically, we set $c_{start} = \frac{7}{4}\overline{\alpha(p)}$, and $c_{end} = \frac{7}{16}\overline{\alpha(p)}$ (the constants can be used to control size and appearance of the fade-out but we kept the values constant in all examples). We fuse this result with the line art error by using a max.

Note that this links the fade-out to distance by construction. This is warranted as disocclusions due to observer position are more likely closer to the observer. The fade-out thus covers the disoccluded part, providing a smooth and consistent transition based on local geometry.

Reprojection Mapping Using reprojection, surfaces of objects can be directly compared, and line art can be selectively removed or its intensity reduced in cases of inconsistencies due to small-scale visible geometry that might be sampled inconsistently between both views (Fig. 7).

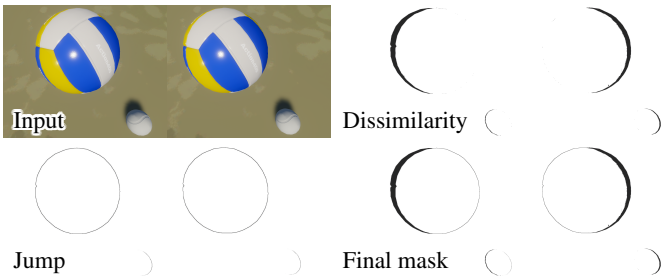

Fig. 8: The reprojection mask excludes dissimilar areas due to occlusion and visible jumps at contours from reprojection.

Reprojection techniques map a pixel from one view to another based on the underlying camera projection [23]. We define a reproject function from view i to j called $R_{i,j}$. For every pixel p_i in view i, we store its fractional pixel position in view j, by projecting the underlying position P_{p_i} via the camera matrix of view j. We then define $D_{i,j}$ measuring dissimilarity between views, and $J_{i,j}$ measuring visible jumps in the opposite view:

$$D_{i,j}(p_i) = R_{j,i}([R_{i,j}(p_i)]) - p_i \tag{9}$$

$$J_{i,j}(p_i, q_i) = (R_{i,j}(q_i) - R_{i,j}(p_i)) - (q_i - p_i) \tag{10}$$

where p_i and q_i are pixels in view i and $[p]$ denotes rounding p to the next pixel position.

$|D_{i,j}(p_i)| = 0$ would mean that the roundtrip of a pixel between both views is perfect. Thus the pixel is visible in both views i and j. The rounding and discretization makes this test imperfect, but the values are expected to be very low.

$J_{i,j}$ is used to find pairs of pixels in view i, that differ significantly after the roundtrip. This happens due to parallax, indicating a grazing angle or discontinuities.

Only if D and J are both close to zero, we can conclude that the location is strongly correlated, and view-dependent effects are negligible. We therefore compute a mask function based on the maximum of D and J:

$$M_{i,j}(p_i) = \max(m(|D_{i,j}(p_i)|), n(\max_{q_i \in N(p_i)} |J_{i,j}(p_i, q_i)|)), \tag{11}$$

where m is a linear map from $[2,4]$ pixels to $[0,1]$, and n is a linear map from $[3,6]$ pixels to $[0,1]$, both bounded on the unit interval. The first interval defines the allowed error for the roundtrip between both views. The second set of constants defines the allowed distance that neighbouring pixels p, q in view i may create in view j. We empirically determined that binocular rivalry is not noticeable when correspondence between lines has a deviation within this range of allowed values. In this situation, an observer seems to be able to match feature lines. A structural investigation of such perceptual phenomena is outside the scope of this paper.

To render the views consistent, we process both views consecutively. Starting arbitrarily with view l, we test for each detected feature line pixel p, to verify that there is a detected feature line in view l in a proximity of two pixels from $R_{l,r}(p)$. If so, we will upper bound $E(p)$ in view l by $M_{l,r}(p) + (1 - M_{l,r}(p)) \max_{q \in \overline{W_5(R_{l,r}(p))}} E(q)$, where $\overline{W_5}$ denotes a 5×5 pixel window. In this way, feature lines are only maintained if they are view consistent. The process for view r is repeated in the same way.

5 Results

We next evaluate the visual quality and runtime performance of our method. We implement our method in Unity Engine 2023.1.10f1 as a Universal Render Pipeline (URP) Renderer Feature using screen-space passes. All stages of our pipeline can be adjusted during runtime. We provide our source code, and include a demo scene and compiled executable as supplemental.[1]

As HMD, we use the HP Reverb G2 Omnicept (G2O), with a render resolution of 3164×3092 per eye, though all HMDs with an OpenXR driver are supported. The XR Device Simulator (XDS) in the Unity XR Interaction Toolkit [30] was additionally tested to isolate benchmarks from driver latencies and VSync timing. The XDS has a render resolution of 1512×1680 per eye.

5.1 Visual Quality

In Fig. 9 we compare the visual quality of five rendering styles: Unity's GGX shading using the default URP Lit shader, a Sobel filter on the

[1] https://github.com/amirzaidi/ieeevr

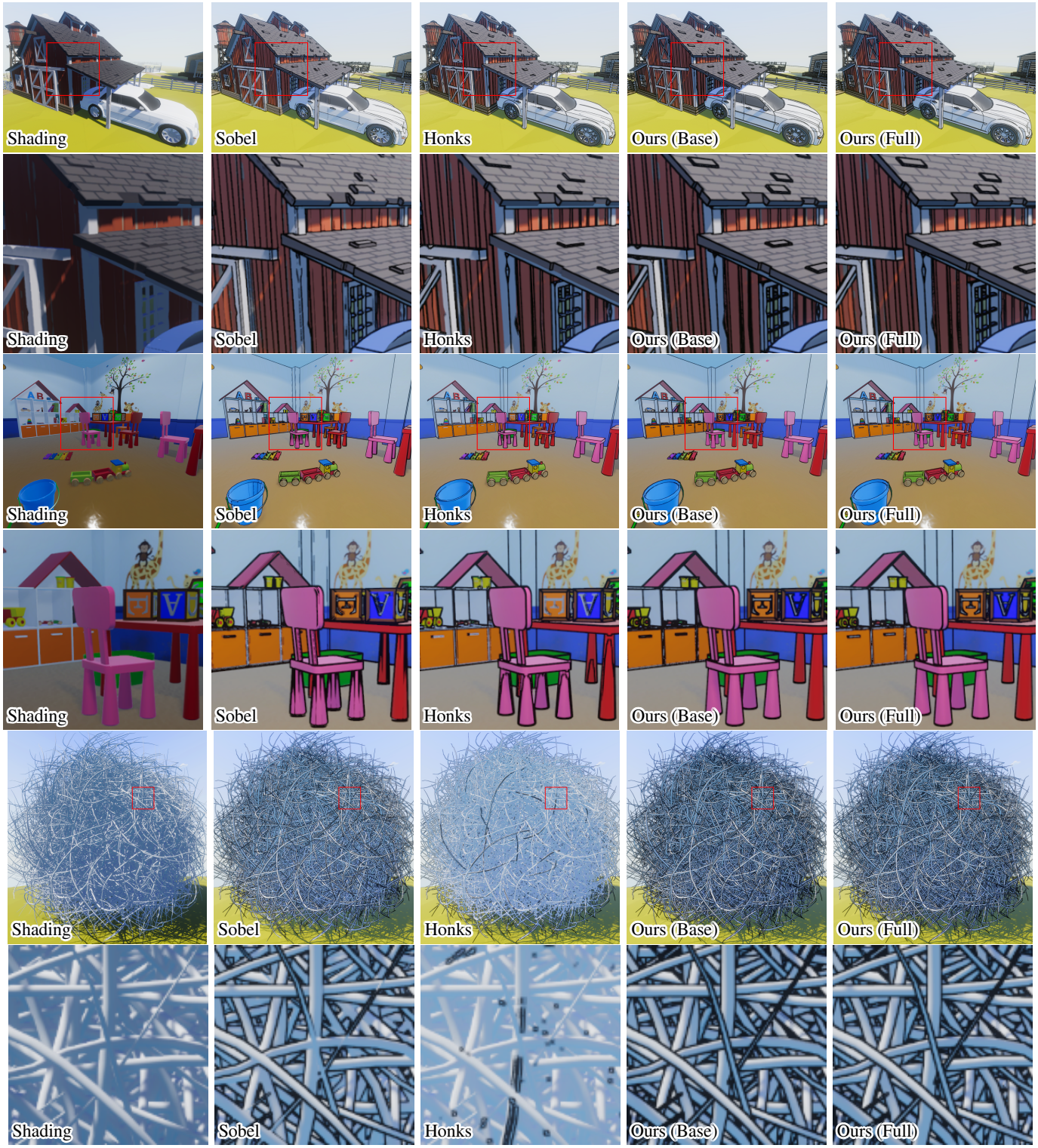

Fig. 9: Top to bottom: POLYGON Farm [25] with a car from CADillac [28], Kindergarten Interior [1], hairball model by NVIDIA [19].

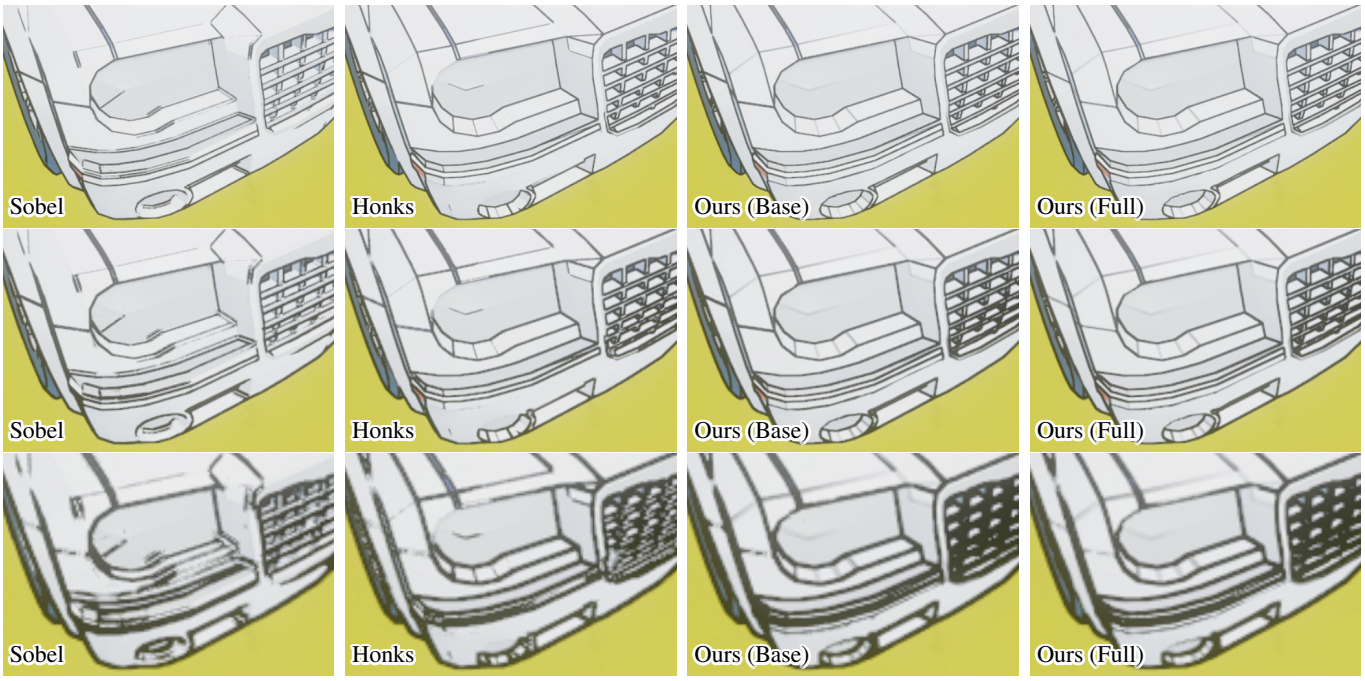

Fig. 10: CADillac car [28]. XDS render scale, top to bottom: 0.75×, 0.50×, 0.25×.

depth buffer, the method of Honks [15] with default thresholds, our base method (Eq. (7)), which we refer to as "Ours (Base)", and our full pipeline including extensions (Sec. 4), which we refer to as "Ours (Full)". All stylisation methods except ours (full) also employ the local-maximum function (Sec. 4) at half strength, to prevent fully black patches from dominating the result. Samples were collected using the XDS running at 1.0× render scale.

As evident in the Hairball example, the Sobel filter either misses lines between close objects or adds lines on geometry where no lines are expected, due to the lack of extra information provided by normal vectors. This filter necessitates a good threshold value, which is challenging as it must work across a range of objects, and the filter must simultaneously detect small distances between micro-geometry and ignore large distances on far-away objects. The method of Honks has default parameters which incorrectly distinguish contours and closed surfaces. Thereby, much of the surface is marked as contour, and removed by our local-maximum function. Our method does not suffer from either issue, as the spherical surface approximation works well across scenes.

In Fig. 10, we compare the four line art styles at lower render scales: 0.75×, 0.5× and 0.25×. At high resolutions, all styles produce similar quality line art. However, our method remains robust at low resolutions. The Sobel filter again suffers from only having the depth buffer as input. The method of Honks performs similarly to ours at 0.50×, maintaining the same set of lines compared to 0.75×, but becomes noisier at 0.25×. While the car grill shows higher contrast in their method, ours preserves contours on minute geometry such as headlights.

5.2 Performance

We evaluate the performance of our method on the XDS and G2O with an Intel Core i7-12700K, NVIDIA GeForce RTX 3090, and 32 GB of DDR5-4400 RAM. As benchmark, we use a scene with multiple objects with sharp edges and measure performance at various render scales. We measure frame time (Fig. 11) and pixel throughput rate (Fig. 12) averaged over 10 seconds. The standard deviation for XDS measurements is below 1 ms. However, the standard deviation for G2O measurements incorporates VSync delays, reaching 5 ms. Over a 10-second average, however, the mean of G2O measurements follows the performance trend in the XDS data. Additionally, we record the GPU time of the methods averaged over 300 frames (Fig. 13), with standard

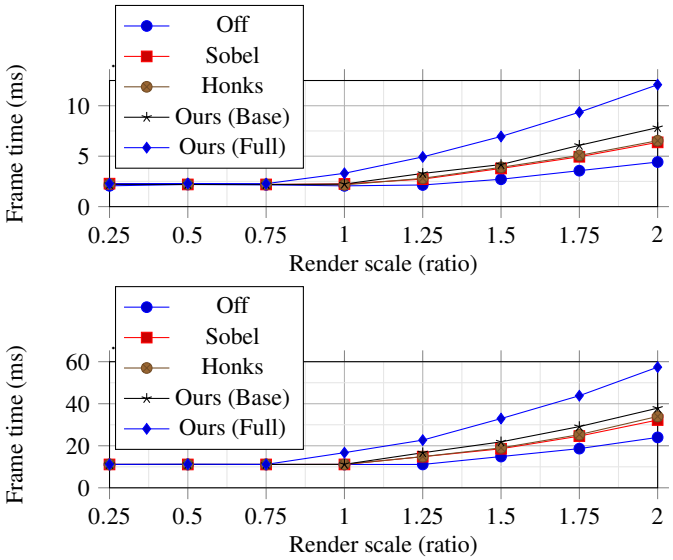

Fig. 11: Top: Average frame times over 10 seconds on the XDS (1512×1680), collected using the Unity delta time measurement. Bottom: Experiment repeated on G2O (3164×3092).

deviation below 1.6 ms even at maximum resolution on the G2O.

As depicted, our full method is costlier than prior methods. However, the base method is comparable to the Sobel filter and the method of Honks. As our test scene is light on geometry, screen-space processing forms the bottleneck. This is a reasonable worst-case, as our method operates in screen space. Further, we note that the pixel throughput of our method follows resolution increases (5.9×10^9 px/s and 2.6×10^9 px/s for base and full respectively); i.e. performances scales linearly with the number of pixels. Thus, if VR target resolutions stay constant in future platforms, maintaining 90 FPS is feasible on consumer hardware.

5.3 Limitations

We next detail shortcomings of our method.

First, spherical LSAs only have zero deviation error for surface patches that are piece-wise representable by spheres. Saddle points can

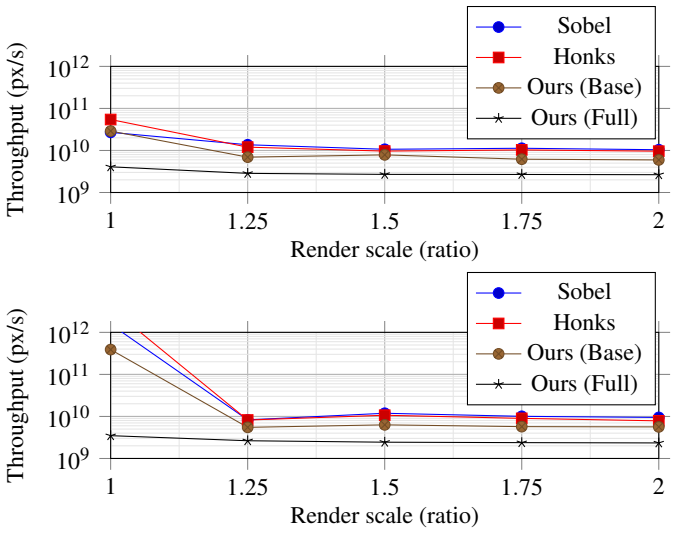

Fig. 12: Top: Relative pixel processing times on the XDS (top) and G2O (bottom), using the same timing data as Fig. 11. At render scales below 1.0 on the XDS, the absolute cost is negligible. At render scales of 1.0 and lower on the G2O, hardware-forced 90 Hz VSync upper bounds the frame rate, resulting in noisy measurements.

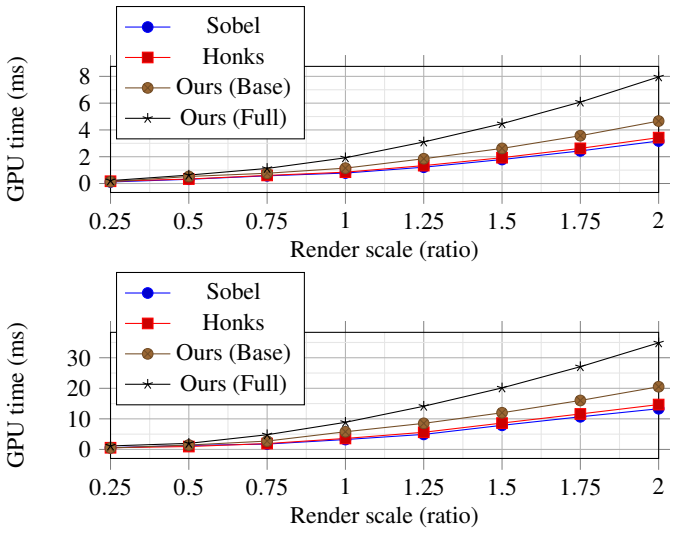

Fig. 13: Top: Average GPU time over 300 frames on the XDS (1512×1680), collected using Unity's Recorder API. Bottom: Experiment repeated on G2O (3164×3092).

cause problems, though our choice of e_{curv} circumvents this. As we use opposites in 3x3 pixel neighborhoods, our world positions necessarily lie on a single plane. Consequently, 3D saddle points simplify to a 2D slice with constant curvature. However, e_{curv} cannot handle planes, which have zero-curvature. We introduce e_{plane} as a workaround, but this necessitates careful thresholding, which is not scale-invariant. We also note that inflection points violate our assumption of near-constant curvature, resulting in deviation error. Thresholding hides inflection points, but these become visible at low resolutions (Fig. 14). Further, the contour fade-out extension is incompatible with (nearly) flat objects, due to the distance between near and far contours becoming extremely large. One heuristic can remove the effect on planar regions: $\max_{q \in \mathrm{W}(p)} \angle(\mathrm{N}_p, \mathrm{N}_q) > \varepsilon$. However, we recommend only enabling this extension in scenes with simple stylized models, to prevent unexpected darkening.

Second, we evaluated a single parameter set for all methods. While our method does not require adjustments for different resolutions or scales, modifying parameters can visually improve the compared methods in some scenarios. We chose a threshold for the Sobel filter that balanced line visibility at a wide range of distances with visual noise. For the method by Honks, we used thresholds from the original implementation. Further, we use scale = 2 to make their method symmetric with identical line thickness to ours, as their default scale only asymmetrically detects discontinuities on the top-right of pixels. All methods can render half-line thickness at the cost of symmetry, but we did not evaluate this configuration.

Last, while our method is robust at low resolutions, all screen space methods inherently operate on limited information. Our method can become noisy around strongly aliased mesh features, notably on brick walls and tiled floors. Additionally, while our method supports normal maps, these necessitate LODs that preserve sharp edges at all distances. Simple bilinear interpolation causes edges to disappear at high resolutions, while nearest neighbour sampling introduces unwanted noise at low resolutions. We rely on Eq. (8) and TAA to improve the quality in aliased and/or normal-mapped regions, and use Sec. 4.2 to ensure noise is stereo-consistent. Our method requires high-precision G-buffers, which is incompatible with variable rate rasterization or MSAA, as inaccurate positions or normals break computations at contours. Consequently, depth-mapped photographs and multi-sample ray tracers are incompatible with our method. However, a regular pixel grid is not required; if $\mathrm{E}(p)$ can be evaluated across a screen space patch by evaluating the fitness of an LSA, our method is applicable. Note that variable rate shading and post-processed methods (e.g. FXAA, SMAA, TAA, and DLAA) are fully compatible with our stylisation, and we use TAA in our experiments unless indicated otherwise.

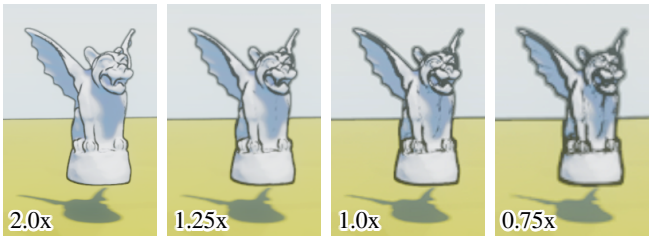

Fig. 14: At low resolutions and large distances, inflection points create a substantial deviation error.

6 User Study

To evaluate the effectiveness of line art on geometric understanding and the quality of our and prior methods, we conducted a user study with three tasks. We recruited 15 participants in the age group 18-76 for task 1, and 20 different participants in the same age group for tasks 2 and 3. The research was approved by and performed under the oversight of the TU Delft Human Research Ethics Committee. Informed consent from the human subjects in the research was obtained using a verbal opening statement. While our ethics committee permitted the study, collecting gender information was strongly discouraged. To our knowledge, participant gender did not influence results. All participants were naive to the goals of the experiments and provided

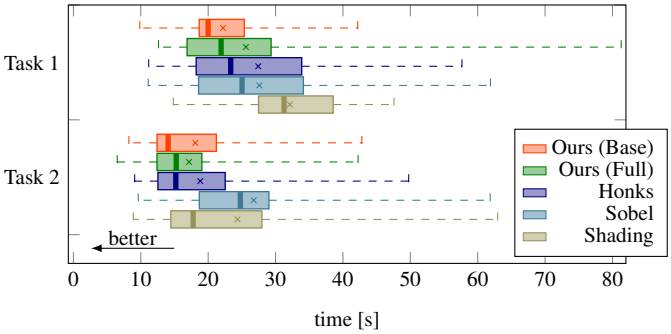

Fig. 15: Time to complete memorization and recognition tasks in five rendering modes, with every trial for every mode being one data point. Box plots show median (marked with a vertical line), lower and upper quartile as well as average value (marked with ×).

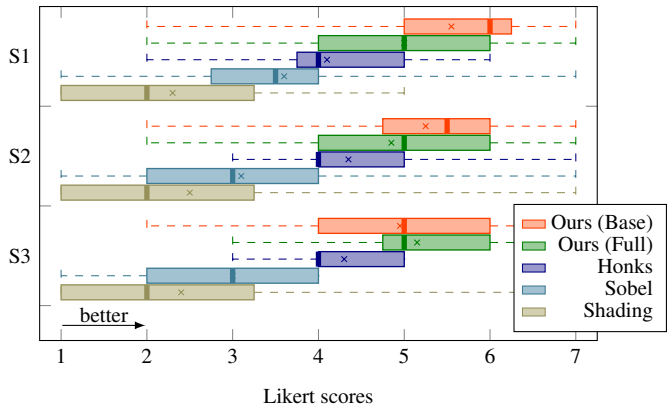

Fig. 16: Rating statistics for S1, S2 and S3 including median (marked with a vertical line), lower and upper quartile as well as average value (marked with ×).

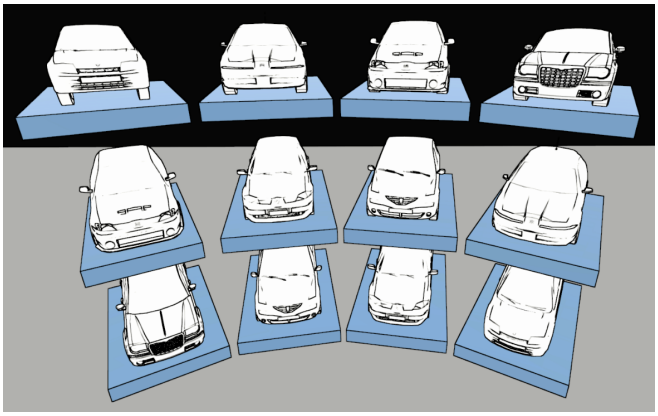

Fig. 17: Evaluating car memorization performance by finding six pairs in a randomized 4×3 grid. Cars are hidden after five seconds.

informed consent without receiving compensation. Additionally, all participants reported normal or corrected-to-normal visual acuity in virtual environments.

Our motivation for user studies is two-fold. First, we evaluate whether line art, regardless of the method, is able to improve both memorization and recognition performance for education. Second, we compare these metrics between low and high-quality line art and, consequently, evaluate the educational value of our method.

6.1 Setup

Task 1. Memorization The first task is an adaptation of the second experiment of Winnemöller et al. (2006) [32], which assessed memory retention with a pair-matching game on a virtual grid of cards. While they compared memorization performance of photographs to an abstracted image style, we compare memorization of shading to line art using six car models with similar silhouettes from CADillac [28]. These models place a strong focus on differences between line art methods around small features of the car models beyond pure silhouette lines. Each model is duplicated, and the six pairs are pseudo-randomly ordered in a 4×3 grid (Fig. 17). Pairs are restricted from occurring in the same row or column. This reduces the timing variance because a perceived structure in the grid order makes the matching game easier to complete. We cycled through the five different styles listed in Sec. 5.1 in a random order. Participants were tasked to complete two practice rounds, followed by ten recorded rounds.

Task 2. Recognition The second task is similar to the experiment of Winnemöller et al. (2007) [31], which tested the recognition performance of abstract shapes rotating and moving on a touch-sensitive board with an overhead projector in different rendering styles. In contrast to their setup, which used objects with large differences between

Task 1	Ours (Base)	Ours (Full)	Honks	Sobel
Ours (Full)	0.0871	-	-	-
Honks	0.0468	0.3972	-	-
Sobel	**0.0123**	0.5470	0.9599	-
Shading	**0.0001**	0.0312	0.1183	0.1109
Task 2	**Ours (Base)**	**Ours (Full)**	**Honks**	**Sobel**
Ours (Full)	0.6217	-	-	-
Honks	0.7941	0.5254	-	-
Sobel	**0.0142**	**0.0084**	0.0568	-
Shading	0.0451	0.0697	0.1586	0.4116

Table 1: p-values for two-tailed t-test between distributions of mean trial time per participant per method. $p < 0.05$ is highlighted after correction by FDR = 10%, $m = 10$ ($p \leq 0.0123$ and $p \leq 0.0142$ for tasks 1 and 2 respectively).

Task 3 S1	Ours (Base)	Ours (Full)	Honks	Sobel
Ours (Full)	0.0873	-	-	-
Honks	**0.0032**	**0.0111**	-	-
Sobel	**0.0015**	**0.0048**	0.1802	-
Shading	**0.0003**	**0.0003**	**0.0017**	**0.0131**
Task 3 S2	**Ours (Base)**	**Ours (Full)**	**Honks**	**Sobel**
Ours (Full)	0.2801	-	-	-
Honks	**0.0232**	0.1336	-	-
Sobel	**0.0006**	**0.0024**	**0.0034**	-
Shading	**0.0009**	**0.0009**	**0.0015**	0.1868
Task 3 S3	**Ours (Base)**	**Ours (Full)**	**Honks**	**Sobel**
Ours (Full)	0.7795	-	-	-
Honks	**0.0232**	**0.0121**	-	-
Sobel	**0.0007**	**0.0002**	**0.0003**	-
Shading	**0.0006**	**0.0004**	**0.0023**	0.0801

Table 2: Wilcoxon signed-rank test p-value between provided answers to three 7-point Likert scale statements on all styles. $p < 0.05$ is highlighted after correction by FDR = 10%, $m = 30$ ($p \leq 0.0232$).

silhouettes, we again use the car models with similar silhouettes from CADillac [28]. 128 floating white cars are shown to the participants, randomly selected from the six car models, which move toward the participant. Furthermore, we randomly placed spheres, cylinders and cubes in the background behind the cars (Fig. 18). Once cars leave the visible area behind the participant, they reappear on the front side of the participant, slightly outside the visible area. During a trial, participants were instructed to find a moving car that matched a stationary car presented in front of them, grab the moving car with their controller, and hold it for three seconds. The participant was then informed whether their choice was correct by a red or green progress bar on their right controller. If the car matched, the style switched, and the next trial began. Otherwise, the participant had to try again until they found a match. Similar to the first task, every participant automatically cycled through a random order for the five different styles listed in Sec. 5.1. Participants were tasked to complete five practice rounds, followed by 25 recorded rounds.

Task 3. Preference The last task collects user preferences regarding the five different styles. Participants were asked, for each style, to rate how much they agreed with three statements on a 7-point Likert scale. They could switch between styles at will, without knowing which style is which. The three statements were:

- S1: *"This style helps me recognize cars."*
- S2: *"This style captures the features of the cars consistently."*
- S3: *"This style is visually pleasing."*

6.2 Results of the User Study

We present the time needed to complete Tasks 1 and 2 in Fig. 15 and preference ratings of Task 3 in Fig. 16. For significance analysis of Tasks 1 and 2, we perform a two-tailed t-test for each pair of methods. Tab. 1 lists p-values for all comparisons between distributions of per-participant mean times. We highlight significant differences where $p < 0.05$ after correction by FDR = 10%, $m = 10$ (two in both tasks). Tab. 2 shows a similar analysis for the preferences according to the

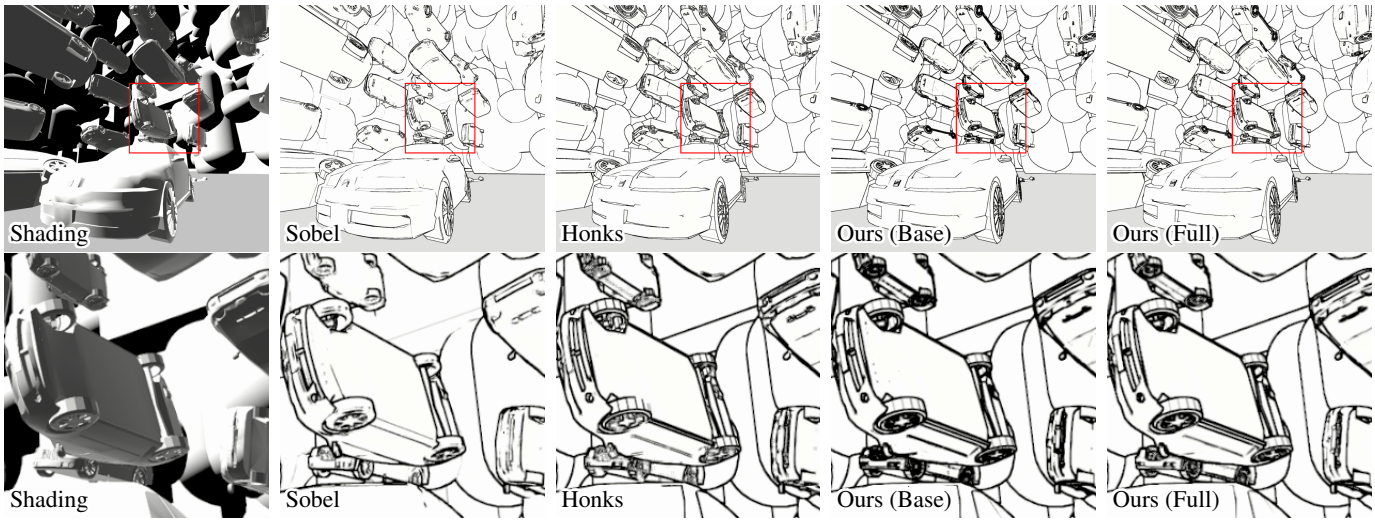

Fig. 18: GGX shading with a strong directional light source and stylisation methods are compared in car identification performance.

statements S1, S2 and S3 in task 3, obtained from a Wilcoxon signed-rank test. Here, we highlight all 22 significant differences between methods with $p < 0.05$ after correction by FDR = 10%, $m = 30$.

Next, we detail our findings. First, prior and new line-art methods can improve performance on Tasks 1 and 3 over shading. In Task 1, while all four stylization methods improved performance upon shading, only ours (base) had a significant improvement. In Task 2, all but the Sobel filter likewise showed some improvement over shading, but no significant improvement was detected. In Task 3, ours (base), ours (full), and Honks are significantly preferred over shading for all statements, and even Sobel was preferred over shading for S1.

Second, feature line quality appears correlated with improved performance on Tasks 1 and 2. Notably, our method demonstrated superior memorization performance over shading; metrics on which prior methods did not show significant improvement. Further, our method demonstrated superior memorization and recognition performance over Sobel. The Sobel filter performs well on Task 1 as all cars are approximately at the same distance. (Fig. 17), thereby providing good results for the selected Sobel threshold. In contrast, Task 2 has cars at varying distances (Fig. 18), creating inconsistencies in detected features between nearby and distant cars, explaining the poor performance of Sobel in this task. Honks had comparable results to our method, indicating that small stylisation differences do not significantly impact results, and the choice of method is up to qualitative preference. Overall, the results do provide strong evidence for the benefits of high-quality line art for both memorization and recognition.

Third, considering participant preferences in Task 3, the distribution of Likert-scale ratings for S1, S2 and S3 indicates that our methods significantly outperform prior work. However, participants rated our full method lower than our base method on S1 and S2, and it likewise had worse performance in Tasks 1 and 2. We speculate that the qualitative improvements in Sec. 4.2 do not translate into additional recognizable features, and participants prefer the noisier output for faster recognition. Participants did rate the full method slightly higher on S3, indicating that they consider the improvements useful for visually pleasing stylisation.

7 Conclusion

We presented LSA Contours, a general method for adding line art to a renderer by detecting geometric discontinuities in screen-space, which requires no world-space preprocessing and supports normal- and bump-maps. Our method runs in real-time at high resolutions and automatically handles stereoscopic inconsistencies. We further proposed the contour fade-out, a selective transition between contours of two different views that counters binocular rivalry. Our user study indicates that our method is preferred over alternative methods, and shows that line art can help with object memorization and recognition metrics. The latter is an important point, as it shows that the development of novel rendering techniques for VR can have an important impact for education or visualization purposes.

Acknowledgements

We offer our gratitude to Mark van de Ruit for assisting with the writing and proofreading, to Richard Grimes and Santosh Ilamparuthi for their ethics committee approval guidance, and to Keenan Crane, NVIDIA and CADillac for publicly releasing the meshes used for evaluation.

Research was supported by the e-DIPLOMA project (101061424)[2], funded by the European Union. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

Supplemental Materials

Source code and a demo application are available as supplemental materials on GitHub at `https://github.com/amirzaidi/ieeevr`, released under a MIT license. Additionally, the benchmarks presented in Fig. 11, Fig. 12 and Fig. 13 can be reproduced using the designated benchmarking scene in the code.

References

[1] 3D Everything. Kindergarten Interior | 3D Interior | Unity Asset Store. https://assetstore.unity.com/packages/3d/props/interior/kindergarten-interior-111197, Mar. 2018. Accessed on 2024-09-19. 5

[2] A. Appel. The notion of quantitative invisibility and the machine rendering of solids. In *Proceedings of the 1967 22nd national conference*, ACM '67, pp. 387–393. Association for Computing Machinery, New York, NY, USA, Jan. 1967. doi: 10.1145/800196.806007 1

[3] E. Bayle, E. Guilbaud, S. Hourlier, S. Lelandais, L. Leroy, J. Plantier, and P. Neveu. Binocular rivalry in monocular augmented reality devices: a review. In *Situation Awareness in Degraded Environments 2019*, vol. 11019, pp. 136–149. SPIE, May 2019. doi: 10.1117/12.2518928 1, 2

[4] D. R. Bukenberger, K. Schwarz, and H. P. A. Lensch. Stereo-Consistent Contours in Object Space. *Computer Graphics Forum*, 37(1):301–312, 2018. doi: 10.1111/cgf.13291 2

[5] P. Bénard and A. Hertzmann. Line Drawings from 3D Models. *Foundations and Trends® in Computer Graphics and Vision*, 11(1-2):1–159, 2019. arXiv:1810.01175 [cs]. doi: 10.1561/0600000075 2

[6] P. Bénard, A. Hertzmann, and M. Kass. Computing smooth surface contours with accurate topology. *ACM Transactions on Graphics*, 33(2):19:1–19:21, Apr. 2014. doi: 10.1145/2558307 1

[2] `https://e-diplomaproject.eu/`

[7] F. Cole, K. Sanik, D. DeCarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh. How well do line drawings depict shape? In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pp. 1–9. Association for Computing Machinery, New York, NY, USA, July 2009. doi: 10.1145/1576246.1531334 2

[8] D. DeCarlo, A. Finkelstein, and S. Rusinkiewicz. Interactive rendering of suggestive contours with temporal coherence. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, NPAR '04, pp. 15–145. Association for Computing Machinery, New York, NY, USA, June 2004. doi: 10.1145/987657.987661 2

[9] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics*, 22(3):848–855, July 2003. doi: 10.1145/882262.882354 2

[10] E. Eisemann, H. Winnemöller, J. C. Hart, and D. Salesin. Stylized Vector Art from 3D Models with Region Support. *Computer Graphics Forum*, 27(4):1199–1207, 2008. doi: 10.1111/j.1467-8659.2008.01258.x 1

[11] Epic Games. Bump Mapping Without Tangent Space In Unreal Engine | Unreal Engine 5.4 Documentation. https://dev.epicgames.com/documentation/en-us/unreal-engine/bump-mapping-without-tangent-space-in-unreal-engine. Accessed on 2024-05-13. 2

[12] S. Grabli, F. Durand, and F. Sillion. Density measure for line-drawing simplification. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pp. 309–318, Oct. 2004. ISSN: 1550-4085. doi: 10.1109/PCCGA.2004.1348362 3

[13] S. Grabli, E. Turquin, F. Durand, and F. X. Sillion. *Programmable Style for NPR Line Drawing*. The Eurographics Association, 2004. ISSN: 1727-3463. 1

[14] D. He, R. Wang, and H. Bao. Real-Time Rendering of Stereo-Consistent Contours. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 81–87, Mar. 2019. ISSN: 2642-5254. doi: 10.1109/VR.2019.8797990 2

[15] R. Honks. Unity Outline Shader Tutorial at Roystan. https://roystan.net/articles/outline-shader/, Feb. 2019. Accessed on 2024-05-19. 1, 6

[16] T. Judd, F. Durand, and E. Adelson. Apparent ridges for line drawing. *ACM Transactions on Graphics*, 26(3):19–es, July 2007. doi: 10.1145/1276377.1276401 2

[17] T. Kaneko, T. Takahei, M. Inami, N. Kawakami, Y. Yanagida, T. Maeda, and S. Tachi. Detailed Shape Representation with Parallax Mapping. In *Proceedings of ICAT*, vol. 2001, pp. 205–208, 2001. 2

[18] Y. Kim, Y. Lee, H. Kang, and S. Lee. Stereoscopic 3D line drawing. *ACM Transactions on Graphics*, 32(4):57:1–57:13, July 2013. doi: 10.1145/2461912.2462001 2

[19] S. Laine and T. Karras. Two Methods for Fast Ray-Cast Ambient Occlusion. *Computer Graphics Forum*, 29(4):1325–1333, 2010. doi: 10.1111/j.1467-8659.2010.01728.x 5

[20] A. Lopez, F. Lumbreras, J. Serrat, and J. Villanueva. Evaluation of methods for ridge and valley detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):327–335, Apr. 1999. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/34.761263 2

[21] L. Northam, P. Asente, and C. S. Kaplan. Consistent stylization and painterly rendering of stereoscopic 3D images. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, NPAR '12, pp. 47–56. Eurographics Association, Goslar, DEU, June 2012. 2

[22] T. Saito and T. Takahashi. Comprehensible rendering of 3-D shapes. *ACM SIGGRAPH Computer Graphics*, 24(4):197–206, Sept. 1990. doi: 10.1145/97880.97901 1

[23] D. Scherzer, L. Yang, O. Mattausch, D. Nehab, P. V. Sander, M. Wimmer, and E. Eisemann. A Survey on Temporal Coherence Methods in Real-Time Rendering. 2011. doi: 10.2312/EG2011/stars/101-126 4

[24] P. Shi, M. Billeter, and E. Eisemann. Stereo-consistent screen-space ambient occlusion. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(1):2:1–2:12, May 2022. doi: 10.1145/3522614 2

[25] Synty Studios. POLYGON Farm - Low Poly 3D Art by Synty | 3D Industrial | Unity Asset Store. https://assetstore.unity.com/packages/3d/environments/industrial/polygon-farm-low-poly-3d-art-by-synty-146192, June 2019. Accessed on 2024-09-19. 5

[26] K. Tanaka and T. Komada. 3D Toon Rendering in 'Hi-Fi RUSH'. https://gdcvault.com/play/1034330/3D-Toon-Rendering-in-Hi, Mar. 2024. Accessed on 2024-05-14. 2

[27] A. Thibault and S. Cavanaugh. Making Concept Art Real for Borderlands. https://stylized.realtimerendering.com/#borderlands, July 2010. Accessed on 2024-05-14. 1, 2

[28] E. Toropov and J. Moura. CADillac. June 2019. doi: 10.1184/R1/8262593.v3 5, 6, 8

[29] Unity Technologies. Unity - Manual: Normal map (Bump mapping). https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html. Accessed on 2024-05-13. 2

[30] Unity Technologies. Unity - XR Interaction Toolkit 3.0.3. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/, Feb. 2024. Accessed on 2024-05-19. 4

[31] H. Winnemöller, D. Feng, B. Gooch, and S. Suzuki. Using NPR to evaluate perceptual shape cues in dynamic environments. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, NPAR '07, pp. 85–92. Association for Computing Machinery, New York, NY, USA, Aug. 2007. doi: 10.1145/1274871.1274885 2, 8

[32] H. Winnemöller, S. C. Olsen, and B. Gooch. Real-time video abstraction. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pp. 1221–1226. Association for Computing Machinery, New York, NY, USA, July 2006. doi: 10.1145/1179352.1142018 1, 2, 8

[33] L. Yang, Y.-C. Tse, P. V. Sander, J. Lawrence, D. Nehab, H. Hoppe, and C. L. Wilkins. Image-based bidirectional scene reprojection. *ACM Transactions on Graphics*, 30(6):1–10, Dec. 2011. doi: 10.1145/2070781.2024184 2