Real-Time Pen-and-Ink Illustration of Landscapes

Liviu Coconu*
ZIB / University of Konstanz

Oliver Deussen
University of Konstanz

Hans-Christian Hege
ZIB

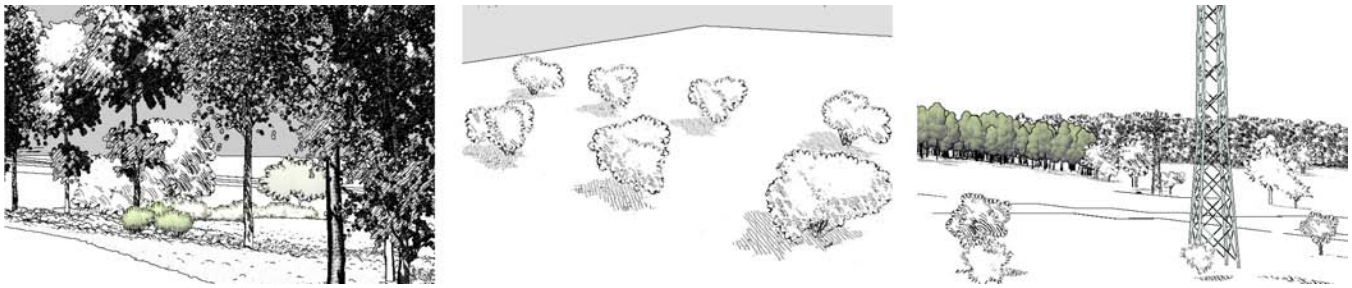


Figure 1: Different landscape illustrations.

Abstract

Non-photorealistic rendering has been proven to be particularly efficient in conveying and transmitting selected visual information. Our paper presents a NPR rendering pipeline that supports pen-and-ink illustration for, but not limited to, complex landscape scenes in real time. This encompasses a simplification framework using clustering which enables new approaches to efficient and coherent rendering of stylized silhouettes, hatching and abstract shading. Silhouette stylization is performed in image-space. This avoids explicit computation of connected lines. Further, coherent hatching of the tree foliage is performed using an approximate view-dependent parameterization computed on-the-fly within the same simplification framework. All NPR algorithms are integrated with photorealistic rendering, allowing seamless transition and combination between a variety of photorealistic and non-photorealistic drawing styles.

Keywords: non-photorealistic rendering, line art, real-time hatching, level-of-detail, complex plant scenes, outdoor scenes

1 Introduction

Real-time visualization of landscapes in 3D has always been a challenging task and is still an active field of research, mainly due to the huge geometric complexity of outdoor scenes. With recent advances in hardware and software, the interactive, photorealistic navigation through impressively large and complex virtual landscapes is possible [Deussen et al. 2002; Behrendt et al. 2005] and future developments will surely break the actual computational limits.

However, as well as in other areas of computer graphics, there are many applications for which photorealistic rendering may not always be the best choice. As examples, in landscape planning and architecture the challenge is not only the computational, but often also the visual complexity of the scenes, making them hardly intelligible. Here, non-photorealistic techniques can be particularly efficient in conveying and transmitting selected visual information. Free from the constraints of photorealism, a controlled, meaningful simplification and structuring of the scene can be performed, leading to comprehensible renditions.

This paper presents an NPR rendering pipeline that supports real-time, hardware-accelerated pen-and-ink illustration of complex landscapes consisting of many detailed 3D plant models. Rather than being replaced, photorealism is tightly integrated with a wide palette of illustration elements, allowing seamless transitions from realistic representations to abstract sketches for maximum flexibility in practice.

Due to the high complexity involved, established NPR algorithms cannot always handle such scenes. The usually most important component of an illustration is represented by silhouette and contour lines. Our contribution is a technique for fast silhouette detection and stylization that, in contrast to previous approaches, avoids the explicit computation of connected contour lines. Silhouette style patterns are defined as two-dimensional textures and can be combined with individual leaf contours.

Another important contribution of the paper is a novel method for real time hatching. The technique introduced by [Praun et al. 2001] does not work well for non-compact, highly fragmented objects such as trees, because it requires a smooth object-space parameterization. On the other hand, image-space hatching such as in [Lake et al. 2000] suffers from the so-called shower-door effect: strokes remain fixed on the screen rather than following the object surface. Our solution is based on a combination of dynamic object space parameterization with image-space, view dependant parameterization. It simultaneously offers much better spatial coherence than the object-space approach and greatly reduces the shower-door effect compared to image-based hatching. The method can be used with standard tonal art maps of [Praun et al. 2001] and requires no additional model information or parameterization.

All NPR algorithms are driven by an object-space abstraction mechanism that works by clustering leaf primitives together into so-called higher-level primitives. This enables abstract coloring, as well as other interesting NPR effects.

*e-mail: coconu@zib.de

NPAR 2006, Annecy, France, 05–07 June 2006.
© 2006 ACM 1-59593-357-3/06/0006 $5.00

2 Related Work

Significant progress has been achieved recently in the area of non-photorealistic rendering, but existing algorithms are usually limited to specific applications, mostly technical illustrations of compact objects. A number of works have been presented in recent years that focus on plant models and outdoor scenes.

Smith was one of the first authors describing fractals and formal plant descriptions [Smith 1984] for computer graphics. Besides other models he generated a computer-generated cartoon tree. The branches of this tree display disks to represent leaf clusters. Reeves and Blau [1985] in their famous work on rendering plants implemented a similar form using small discs for the production of their (realistic) trees. Sasada [1987] uses tree sketches in an architectural environment. For the rendering of his trees he uses computer-generated tree skeletons, the renditions are then projected as textures onto billboards.

In [Salisbury et al. 1997] a tree sketch is modeled using so-called stroke textures that were introduced in [Salisbury et al. 1994; Winkenbach and Salesin 1996]. In this approach the directions of the foliage and trunk silhouette are modified by a given vector field, and by an additional gray-scale value image. The so-called difference-image algorithm is used, which places textures or strokes in the resulting image until the differences in the gray-scale values between the given and the resulting image are sufficiently small.

Kowalski et al. [1999] introduce a method to illustrate plant scenes in the style of two well-known authors of children's books. In contrast to the already mentioned procedures of Salisbury et al. [1997], they deal with an automatic method that also uses a 3D model as its basis. For rendering the image, the authors apply a multilevel algorithm, which in the first step illustrates the scene conventionally. The gray-scale values of the created image are the starting point for the placement of so-called "graftal textures", which are positioned at those places that appear dark in the initial image using the same difference-image algorithm mentioned above.

Deussen and Strothotte [2000] present another image-based method for rendering tree models. The depth differences of the pixels in the image are analyzed and silhouettes are drawn only if the difference is above a given threshold. Using so-called drawing primitives they are able to achieve a sufficient degree of coherence in their images. However, hatching and silhouette stylization are not explicitly addressed. Our work builds on their approach of drawing primitives and depth differences: while using similarly efficient image-based techniques, our high-level primitives allows for significantly more versatile visual composition.

In [Wilson and Ma 2004] several methods for hatching tree objects are presented using a set of two-dimensional buffers. In this work nice images have been created, however, no coherence is maintained which prohibits the results from being used in animations. An interesting but computationally costly method for computing silhouette lines of complex objects is presented in [Sousa and Prusinkiewicz 2003]. Another aspect was covered in [Xu and Chen 2004]. In contrast to the above methods here real-world data was obtained by scanning and methods for the abstraction and stylization were applied.

Silhouette stylization has also been explicitly addressed ([Markosian et al. 1997; Hertzmann and Zorin 2000]). Mostly, an explicit line description is needed in order to apply a silhouette pattern, which are hard to obtain efficiently for complex landscape scenes. Kalnins et al. [2003] develop a coherence-improved arc-length parameterization. In [Northrup and Markosian 2000] the 2D projection of the silhouette edges is used to merge edge segments into long strokes. Hardware-based methods like in [Gooch et al. 1999] and [Raskar and Cohen 1999] admit only minimal style control. The "loose and sketchy" technique of Curtis [1998] admits a limited form of stylization by displacing the original, image-based silhouette Our stylization approach uses a similar idea, but runs in real-time and allows for arbitrary 2D textures for stylization.

3 Rendering Framework

Our algorithms work on a level-of-detail description for complex plant scenes that was provided by Deussen et al. [2002]. The data is given as a set of plant models distributed over a terrain model. For near views, a detailed polygonal model is used for each plant. As distance increases, a level-of-detail data structure is used for efficient rendering by using disks for the leaves and lines for branches. This mechanism is also used to achieve simplified renditions, by increasing the size of the leafs, similar to the drawing primitives of [Deussen and Strothotte 2000].

In contrast to previous approaches, the user is provided with a continuous palette of real-time rendering styles that can be simultaneously used for different parts of the scene. Rendering is driven by several parameters, depicted as axes in Fig. 2 and discussed in detail in the remainder of the paper.

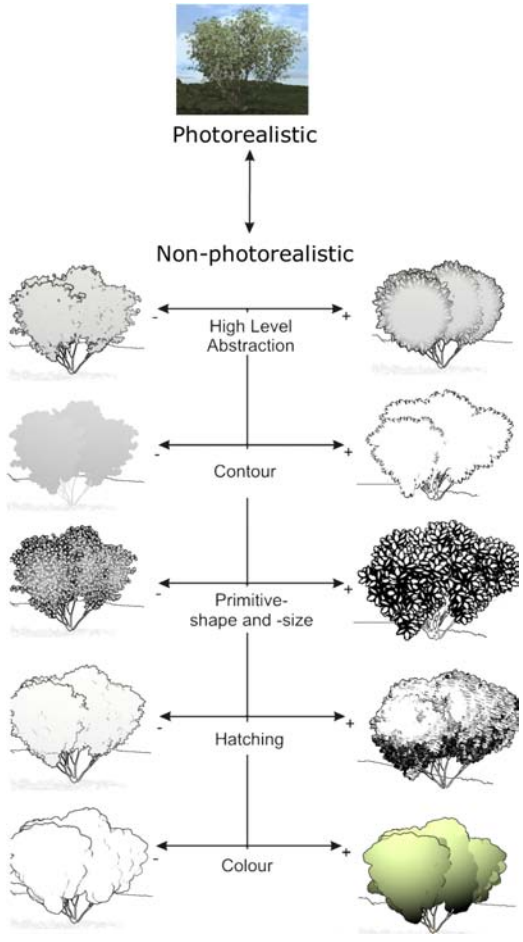


Figure 2: An overview of rendering styles. The axes correspond to sliders in our interactive editor.

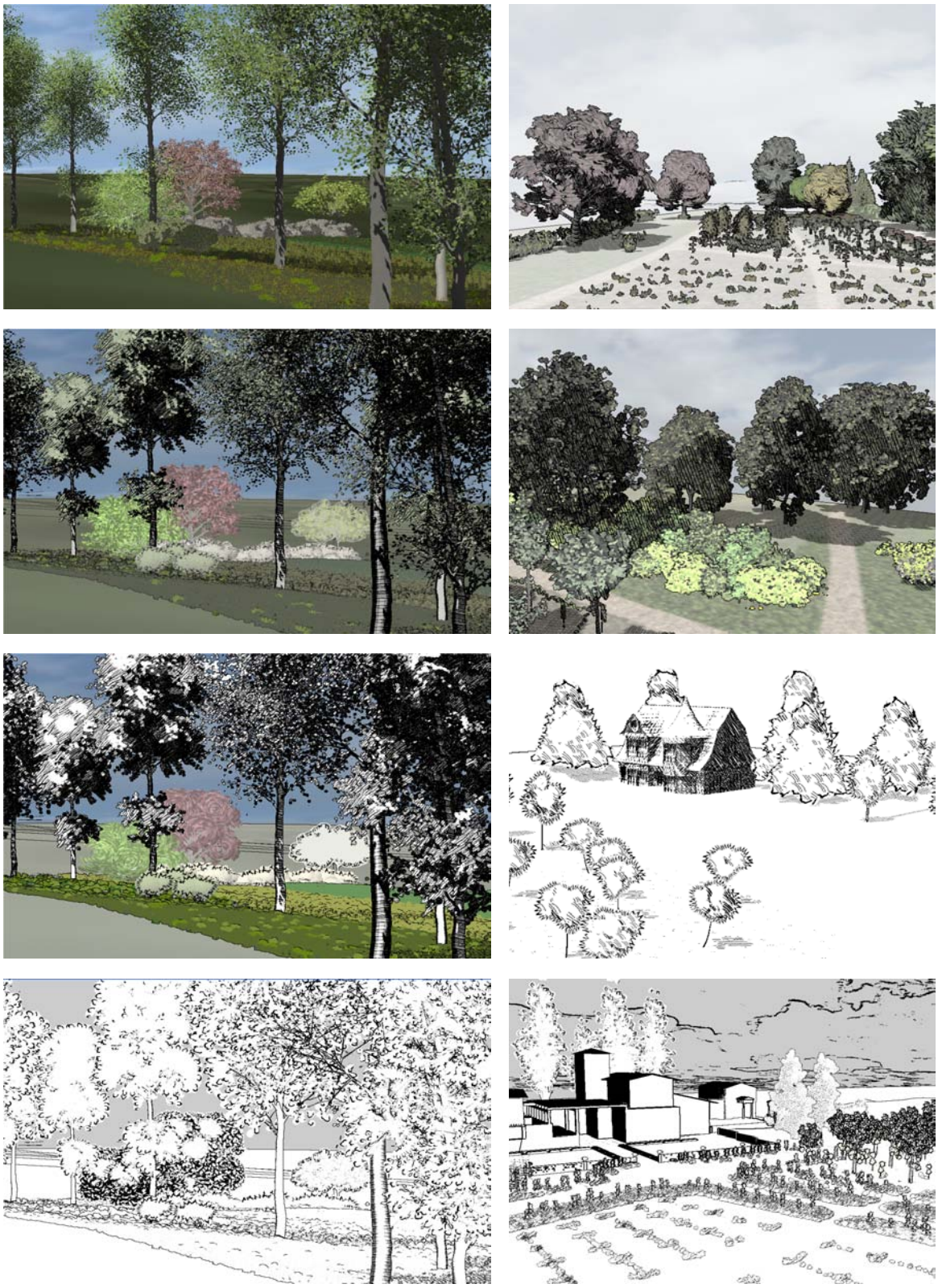


Figure 3: Left column, from top to bottom, continued abstraction of a landscape: photorealism, half-NPR, full NPR, silhouettes and hatching only. Right column: several other abstract renditions, from top to bottom: Van Gogh look (abstract, elliptical leafs around the HLPs), pointilistic style, minimalistic drawing, combination of different styles.

4 Axis "Higher Level Abstraction"

While a certain visual simplification can be achieved by manipulating the size and shape of the leaf primitives and choosing appropriate level-of-detail like in [Deussen and Strothotte 2000] (see Section 6), the degree of abstraction is limited by the underlying geometric model for a given primitive size. In order to allow for more abstract renditions, we employ a more general mechanism which works by clustering foliage primitives (leaves) into larger and more abstract 3D-shapes which we call higher level primitives (HLP), such as spheres.

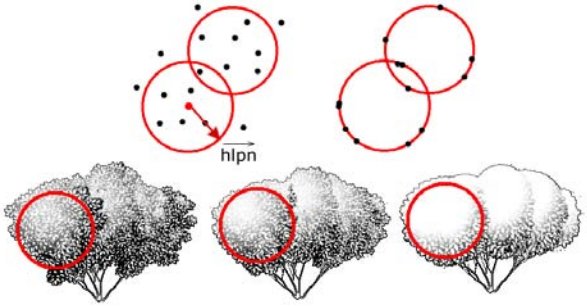


Figure 4: Top: Leaves (represented as dots) are grouped into spherical HLPs and associated a HLP normal. Bottom: Different degrees of abstraction obtained by displacement of leaves, shading and detail reduction according to higher level primitives, for blending factors 0, 0.5 and 1.

The algorithm can be regarded as a re-modeling process: we fit higher-level primitives (HLP) in the tree foliage. We use a greedy clustering algorithm: leaf positions are successively checked against the current HLP set, initially void. If a leaf can be added to an existing HLP without increasing its size over a user-defined threshold, it is added to that HLP, otherwise a new HLP containing the leaf is created (currently, we use spheres as HLPs, defined by a center and a maximum radius r, as Fig. 4 shows). For each leaf, we define and store its corresponding *HLP normal* as the vector from the HLP center to the leaf.

During rendering, the leaves can be re-positioned according to the higher level primitives. We blend between the original leaf position and its projection onto the HLP along the HLP normal, which allows seamless control over the degree of abstraction (Fig. 4). A nice contour effect is achieved by re-orienting the leafs to point away from the the higher-level primitive.

5 Axis "Contour"

5.1 Contour Detection

Because object-space silhouette detection is prohibitive in terms of computational complexity for our complex target scenes, an image-space approach for contours has been used. The general approach to image-based contour detection is to render depth, normals and color g-buffers [Saito and Takahashi 1990] in a first step, then detect discontinuities in these buffers. For plants, only depth discontinuities are used, similar to [Deussen and Strothotte 2000], because discontinuities of first and second order derivatives or normals are unusable for the non-compact foliage of plants. We also found color-derived contours rather visually unpleasing.

A conventional edge detection filter is used for contour detection, evaluating a pixel intensity $p_0 \in [0,1]$ as the sum of the normalized depth differences to its eight neighbours in a 3 x 3 neighborhood, controlled by a threshold t:

$$p_0 = sat\,(0.5 * \sum_i \lceil sat\,(d_i - d_0 - t)\rceil) \qquad (1)$$

where $sat(x) := max(0, min(1,x))$ and d_i are the depths in a 3×3 pixel neighborhood. We use *signed* depth differences in order to obtain 1-pixel thick contours. The values are passed to a further filter which removes isolated pixels by checking the 3x3 neighborhood average (a contour pixel should have at least 2 neighbors)

5.2 Stylization

Although fast and easy to compute, conventional image-based contours have a major drawback: the results are unconnected pixels, thus the potential for stylization is very limited. We extend image-based contours with a novel approach to stylization that applies user-defined 2D-textures onto silhouettes and avoids the requirement of explicit silhouette description.

The main difficulty that has to be addressed is finding a parameterized 2D support for the silhouette textures: one texture coordinate along the contour, u, and the other perpendiculat to it, v. We want to apply the style texture as a post-rendering step in image-space (just after silhouettes are computed) and rely on the 1-pixel wide detected contours as reference. We construct an approximate parameterization, using additional information stored in the g-buffers.

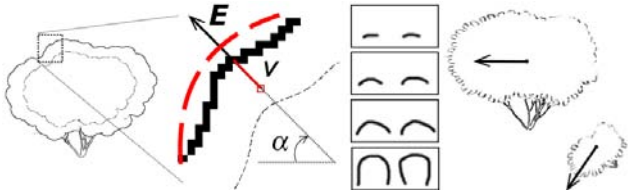


Figure 5: a) Parameterization in image space: 1-pixel contours are expanded inwards (the v coordinate). The parameterization along the silhouette is aproximated with the length of the arc described by α on the HLP projection (the dashed line). b) 4-channel SAM (silhouette art map) and c) zooming and rotation behaviour for this SAM : the per-object 2D orientation vector is depicted, which is the reference for α.

A suitable direction for v should satisfy two important requirements: it must be roughly perpendicular to the contour and it must vary smoothly across the contour to avoid sudden "jumps" in the texture. The image plane projection of HLP normal defined in the previous section has been proven to be a good choice. During scene rendering, we store this 2D vector $\vec{E}$ at each pixel in the g-buffers. To compute the v coordinate at a pixel, the nearest contour pixel is searched along $\vec{E}$, starting from the current pixel. The v coordinate is then the distance to the contour pixel. If no contour pixel is found within a fixed interval (p pixels), we skip the contour rendering for the current pixel - this means that the extent of the search along $\vec{E}$ determines the width of the silhouette support that is expanded inwards (see Fig. 5a).

For the u coordinate along the contours we also use an approximation, because the exact computation along the 1-pixel wide detected

contours is a difficult task. Instead, the length of the path is approximated by the corresponding path on the HLP image-space projection (Fig. 5a) For spherical HLPs, the same vector $\vec{E}$ is sufficient to compute its corresponding angle α described in 2D. The length of the arc described by α is scaled according to the distance, so as to match the v coordinate (which is computed in pixels):

$$u = \frac{c\,\alpha\,r}{d} \tag{2}$$

where r is the HLP radius and c is a user parameter that controls the aspect ratio (the stretching of the texture along the silhouette). In this way, as the camera moves towards the object, the texture is not stretched, but rather new "strokes" are added as u increases(Fig. 5c). Because u is computed in image space, the texture will not follow the rotation of the object (shower door effect). We can partially compensate this by maintaining a 2D orientation vector per object, $\vec{V_O}$, that follows the apparent rotation of the object in image space: $\vec{V_O}$ is initialized to (-1,0) and actualized each frame according only to the rotation of the camera around the view direction (Fig. 5c). Then, the angle α is computed relative to $\vec{V_O}$.

Compared to more elaborate solutions, like in Kalnins et al. [2003], which allows for a trade-off between 2D and 3D coherence, our approach can only maintain 2D arc-length coherence that looks well for panning, zooming and rotation around the view direction. For more complex transformation, texture may slide along the silhouettes - a common coherence problem for NPR animation. Since u is computed relative to a HLP, the continuity of the contour parameterization is limited to that HLP - texture discontinuities will appear at HLP boundaries. Due to the path length approximation, the uniformity of the parameterization along the contour depends on how well the (spherical) HLP approximates the respective patch of the object. These are, however, far less disturbing effects than texture sliding. All these effects can be seen in the accompanying video.

In order to define the style, a 2D texture structure similar to the tonal art maps (TAM) of [Praun et al. 2001] is used, which we call silhouette art map (SAM). There are two differences compared to the TAMs: 1) because the support has a fixed size on the screen, mip-mapping is no longer needed and 2) each channel of the SAM can contain a different silhouette style, without imposing a nesting property like for TAMs. Using multiple channels (usually 4), the style can be changed along the silhouette, by selecting the desired channel, i.e. style, according to different parameters, like shading (the tree in Fig. 5c uses thicker silhouettes for darker shading, giving a hint on the lighting).

In contrast to previous image-based techniques that only allow implicit stylization of contours (like the various drawing primitives of Deussen and Strothotte [2000]), our approach leverages explicit stylization in form of user-defined textures. Compared to methods that explicitly render strokes like Kowalski et al.[1999], there are some limitations in our approach. First, because the stroke texture are rendered in a single pass on the support given by the 2D-expanded contours, strokes belonging to different silhouettes cannot overlap. A possible extension of the technique would be the use of multiple rendering passes to achieve overlapping strokes . Another limitation is that silhouette art maps cannot be transparent, because the post-rendering processing cannot recover the invisible surfaces that would become visible at transparent pixels. Again, multiple rendering passes are a partial solution: we consider this as future work. On the other hand, our method operates in image space taking full advantage of hardware acceleration and thus it can handle very complex scenes.

6 Axis "Primitive Size and Shape"

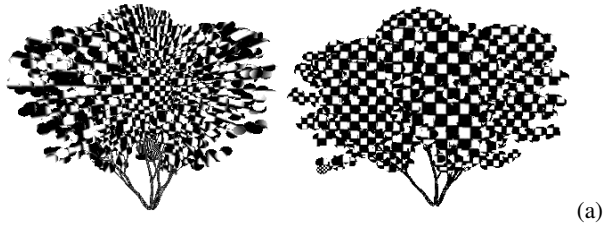
The smooth transition from realistic renditions of a scene to abstract representations requires the ability to change the size and shape of the foliage primitives (trunk and branches maintain a relatively unchanged appearance). For shape control, we construct a special alpha-matted texture for leaves and alpha tests: depending on the alpha threshold, the outline changes from the original photorealistic texture to an abstract shape. In Fig. 6 this procedure is shown for the transition into a disc, but also other shapes are easily achievable by using different alpha maps. We generate the alpha map by radial extension of the original alpha map of the leaf. The individual outline of the leafs can be rendered for "visual agglomeration"-style rendering.

Figure 6: Leaf shape variation encoded in the alpha channel of the texture. From left to right: original leaf texture alpha, radially expanded alpha, shapes obtained with alpha thresholds 0.7, 0.55, 0.4, and 0.25.

7 Axis "Hatching"

Real-time hatching techniques have been developed in both object and image space. Given the efficiency requirements, we follow the tonal art map (TAM) approach of Praun et al. [2001], where hatch strokes are stored at different discrete tone levels in textures, instead of being rendered individually. Smooth tone transition is achieved by interpolation between the levels. In order to maintain hatch density and width, several mipmap levels of each tone are used. For coherent switching, a nesting property is imposed both among different tones and among the mipmaps to avoid popping artifacts.

The approach of Praun et al. [2001] relies on a smooth object-space parameterization to map the texture onto an object. Unfortunately, such parameterizations are ill-suited for highly fragmented objects like the foliage of a tree, where the spatial coherence of the texture is broken, as the example in Fig. 7a shows. Another class of techniques would be image-space hatching, which instead suffers from severe temporal coherence problems ("shower door").



(a)

(b)

Figure 7: Mappings of a checkerboard texture on a tree surface: (a) an object-space spherical mapping - texture coordinates are computed using the projection onto a parameterized bounding sphere. The spatial coherence of the texture is compromised, except for the central region, where the different fragments of the object (leaves) align perpendicular to the camera direction. The effect is worsened by the fact that leaves are always drawn as facing the viewer. (b) our view-dependent mapping using HLPs.

We address these issues with a novel, *view-dependent* parameterization, based on the idea to combine the good spatial coherence properties of image-space parameterization with the temporal coherence of object-space techniques. Only the plant foliage will be considered, as trunk and branches can use the conventional object space parameterization. The idea is similar to the silhouette stylization: we use the partition of the object into high-level primitives (see the re-modeling process in Section 4) and attempt to construct a parameterization for each HLP patch.

A simple object-space parameterization can be obtained by projecting onto the HLP surface and using a parameterization of the HLP, in our case a sphere. We parameterize each hemi-sphere as in Fig. 8a and thus associate each 3D point with the 2D coordinates $(S_u, S_v) =: S_{uv}\ (\phi, \theta) \in [0,\pi) \times [0,\pi)$ of its projection onto the hemisphere it belongs to.

$$S_{uv}(x,y,z) = (m\frac{\phi}{\pi}, n\frac{\theta}{\pi}) \tag{3}$$

where and n, m are the number of texture tiles along u and v, respectively. Such a parameterization exhibits severe coherence problems described above (Fig. 7a), with the exception of the vicinity of the projection of the center. We take advantage of this property and use the spherical parameterization to compute the texture coordinates associated to the viewing vector $\vec{R}$ from the HLP center C to the current eye position E (see Fig. 8). We call $\vec{R}$ the reference vector and the corresponding texture coordinates the reference texture coordinates.

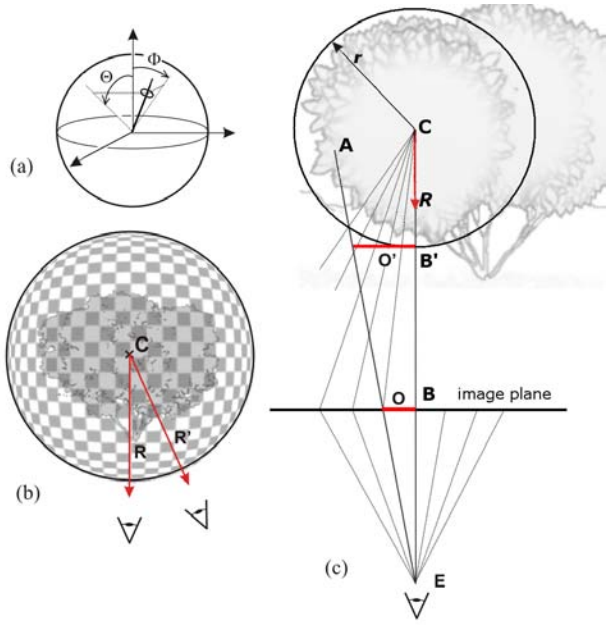


Figure 8: (a) hemi-spherical parameterization; (b) actualization of reference texture coordinates, and (c) offset computation.

In order to avoid the discontinuities of the spherical parameterization defined above, we use it in a view-dependent fashion by defining it centered in the current reference vector $\vec{R}$ (Fig. 8b). As the camera position changes, the reference vector $\vec{R}$ changes to $\vec{R}'$. We store the reference texture coordinates per object and update them according to the camera change by adding the offset of $\vec{R}'$ relative to $\vec{R}$ through the parameterization S:

$$R'_{uv} = R_{uv} + S_{uv}(\vec{R}') - S_{uv}(\vec{R}), \tag{4}$$

where S is the hemispherical parameterization centered in $\vec{R}$, thus $S_{uv}(\vec{R})$ is actually 0. In this way, $\vec{R}$ always remains in the central area of the hemisphere (Fig. 8b) - assuming small frame-to-frame changes in the orientation of the object.

The second, image-space component of the texture parameterization for an arbitrary point A of the object is an offset O relative to the reference vector/texture coordinates. As shown in Fig. 8c, the offset is first computed in the normalized device coordinate system (image plane), then scaled such as to match the reference texture coordinate units and added to the reference coordinates. We accomplish that by projecting the offset back to the camera coordinate system at the HLP surface (O') and approximating the corresponding angle on the sphere (division by the HLP radius):

$$O_{uv} = \frac{1}{r}[\frac{(P\cdot M\vec{A})_{xy}}{(P\cdot M\vec{A})_w} - \frac{(P\cdot M\vec{C})_{xy}}{(P\cdot M\vec{C})_w}]\,\frac{d}{d'}/(\frac{\pi}{m},\frac{\pi}{n}) \tag{5}$$

$$T_{uv} = R_{uv} + f(\vec{A})\,O_{uv}, \tag{6}$$

where r is the radius of the HLP, M is the modelview matrix, P the perspective projection, d, d' are the distances EB and EB', $/$ denotes component-wise division of the 2D vector, and $f(\vec{A})$ is an optional surface function defined in image-space, for now $f(\vec{A}) = 1$.

The orientation of the hatch texture is constant in image space, which implies a shower-door effect when rotating around the camera's viewing direction. This can be compensated by using the same method as for contours, based on the 2D orientation vector $\vec{V_O}$. The texture coordinates computed above are simply aligned with $\vec{V_O}$:

$$\overline{O}_{uv} = Q_{V_O}\,O_{uv}, \tag{7}$$

where Q_{V_O} is the matrix that performs the 2D rotation of the texture coordinates to align the v-axis with $\vec{V_O}$.

Using this parameterization, strokes will lack any parallax effects, since the texture is simply applied in image space in a flat fashion (see Fig. 9a). Alternatively, the surface function $f(\vec{A})$ can be used to avoid the flatness of the hatch textures by incorporating different surface properties. For example, we used camera-space depth as follows:

$$f(\vec{A}) = 1 + c[\frac{1}{r}(M\vec{A})_z - (M\vec{B}')_z]^2 \tag{8}$$

to suggest the shape of the object. As shown in Fig. 9b and c, stroke orientation is still constant, but the distance between them varies with the distance to the camera, creating a paralax effect and a hint of depth.

Just as for contours, the mixture between object-space and image-space parameterizations is not fully temporal coherent: strokes appear to "adhere" to the object's surface only at the projection of each HLP center and there will be some texture sliding as the distance to the center increases. As an analysis of hand-drawn sketches reveals, this approach matches most hatching styles for plants: strokes have a constant hatch direction, in contrast to the 3D-coherent, curvature-aligned strokes that are used for smooth objects. Therefore, there is no obvious way to define a temporally coherent behavior for the former styles anyway. There is a trade-off between temporal and spatial coherence according to the HLP size: with smaller HLPs,

(a) (b) (c)

Figure 9: Effect of different parametrizations on hatching: (a) flat parametrization according to Eq. (5), (b) non-flat parametrization according to Eqs. (8, and (c) non-flat parametrization with Eq. (8), but with the function $f(A)$ discretized in 3 steps.

temporal coherence increases because there are more "adherence" points, but the spatial coherence may decrease, as the stroke support becomes more fragmented. Because texture coordinates are computed relative to each HLP, there will be texture discontinuities across the borders of the HLP patches. These artifacts tend to be hidden by the complex, non-smooth structure of the foliage.

Besides parameterization, a shading model for the generation of hatching tones is also needed. For more pleasant results, the photorealistic local shading can be gradually combined with more abstract shading as described in the next section.

8 Axis "Shading"

Adding features and color to sketchy renditions can be done by using the same source data available for photorealistic rendering (normals, textures). However, more abstract views of a scene are usually desired. In contrast to the attempt of Wilson et al. [2004] to automatically adjusts complexity based on a measurement of the detail, we make use of our HLP abstract representation to explicitly control the appearance (shading, hatching) and detail density of the produced drawing. The HLP normals can be used for shading instead of or combined with the original normals of the model. This results in smoother, more abstract shading of plants. We use a variation of the Phong model, but virtually any local shading model relying on normals can be used to create different effects, such as Gooch shading, for instance ([Gooch et al. 1998]) Following hand-drawn examples, detail (such as leaf contours) is reduced around the HLP center. HLPs also enable other interesting leaf effects, like the "Van Gogh" style in Fig. 3 top-right.

9 Implementation and Results

Most of the algorithms described above can take advantage of modern programmable graphics hardware: they are implemented as vertex and fragment programs (see Fig. 10). The rendering pipeline is split into two parts. In the first stage, the scene is rendered into a multiple buffer with all the attributes necessary for further processing. In the second part, these attributes to construct the different illustration elements, which are combined together according to the parameters specified by the user.

We rendered test scenes on a 2.4 GHz Pentium 4 with nVidia GeForce 6600 graphics, using the multiple render target feature to implement the multi-buffer. Table 1 shows the rendering performance for the relatively complex scenes in Figure 1. It can be observed that frame rates are comparable to traditional photorealistic rendering (using moderately complex local shading), although an accurate comparison is dificult to make because NPR performance

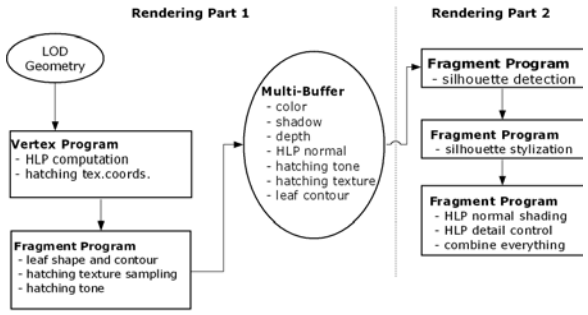


Figure 10: Layout of the rendering pipeline: part one is computing multiple buffers, part two does the final rendering.

heavily depends on the chosen styles. The same level-of-detail mechanism is used, which adapts the number of leaf primitives to the distance to the object. For NPR renditions, the leaf size can be much increased, resulting in less drawn primitives. This more aggresive reduction tends to asympthotically balance out the computation overhead of NPR algorithms for larger scenes. Combining different styles in one view requires the post-processing shaders to be run once for each style. Although the time spent in post-processing is constant for a given screen resolution (25 ms on our system), it can significantly reduce performance if many styles are combined. Constant post-processing time and the level-of-detail technique ensures a total rendering time that is not so much depending of the scene size, allowing the rendering of realistic outdoor scenes in real-time. Faster graphics hardware and a more optimized implementation should significantly improve performance.

Sihlouette stylization is implemented in a post-processing shader. The maximum width of the silhouette support that can be handled is implementation-dependent: we use a maximum of 12 pixels, but this number can be increased at the cost of longer shaders: each additional pixel requires a texture sampling instruction, as well as updating the distance to the origin of the search. For hatching, we preferred to sample the texture in the earlier rendering stage to avoid texture coordinate precision issues.

Fig. 3 shows the same scene rendered in a variety of abstraction levels - the capability of our system to combine several rendering styles in the same view is also illustrated. All renditions are produced with a frame rate of some images per second. We expect the gain in comprehensibility and flexibility to make our framework a viable alternative for the visualization of complex landscapes. Another interesting feature is the user interaction: our system allows the user to paint the silhouette and hatching textures on-the-fly (in a small window), such that the effect of every added stroke can be seen immediately in the scene.

		PR		NPR	
Fig.	# polygons	% lod	fps	% lod	fps
1(b)	1,113,944	25.0	32.6	10.30	23.4
1(a)	93,134,464	2.82	3.1	1.13	5.4
1(c)	378,106,940	0.1	8.1	0.04	5.6

Table 1: Rendering performance for different scenes at a resolution of 800x600. The *lod* column indicates how much of the total geometry of the scene is actually rendered.

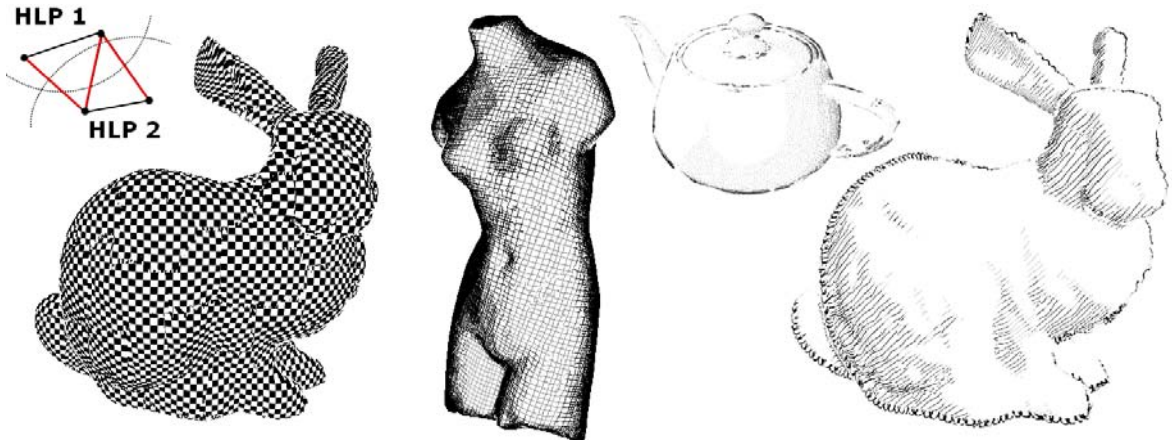


Figure 11: Silhouettes and hatching patterns on smooth surfaces. The models are split along edges that span across different HLPs to avoid incorrect interpolation. The teapot and the bunny model on the right also show stylized, shading-dependent silhouettes (charcoal and a "curly" texture, respectively.)

10 Extension to smooth objects

An interesting experiment has been to apply our stylization and hatching method on regular, smooth 3D models using the same HLP re-modeling. For hatching, because the spatial coherence problems no longer apply, we can also compute the offset O_{uv} in 3D using the hemi-sphere parametrization of the HLP (Eq. 3 and Figure 8). Equation 5 becomes:

$$O_{uv} = S_{uv}(\vec{A} - \vec{C}). \tag{9}$$

To approximately account for the shape of the object, which is should be suggested by the direction of strokes, we use the surface function:

$$f(\vec{A}) = length(\vec{A} - \vec{C}). \tag{10}$$

There is one more issue to be addressed: the vertices of a triangle may belong to different HLPs, which leads to incorrect interpolation of texture coordinates across two patches (Fig. 11). Performing all computations per pixel is expensive, so this has been solved by duplicating those vertices such as the vertices of each triangle point to the same HLP attributes. In Fig. 11 some illustration examples are shown. The stroke direction suggests the local shape. While this is not a true alignment to curvature or other direction fields, it still yields usable results. Sliding and discontinuity artifacts tend to be significantly more noticeable than for non-compact objects - depending also on the applied stroke textures (less obvious for irregular patterns). Again, the errors depend on how well the HLP model approximates the original model.

11 Discussion and Future Work

We have presented a framework for real-time illustration of complex landscape scenes that provides the user with the ability to control the appearance and drawing style of the scene by merely changing a few parameters. Re-modeling with high-level primitives enables meaningful simplification as well as novel on-the-fly parameterization techniques for efficient stylization of silhouettes and real-time hatching of plants with spatial stroke coherence and reduced "shower door" effect. Moreover, photorealism and non-photorealism are no longer separated, but naturally integrated in the same rendering style palette.

Currently, there are a few drawbacks and several further research directions we would like to explore in the future. The higher level primitives we introduced can be systematically extended to other 3D shapes than spheres (ellipsoid, cylinder, etc.), in order to better approximate different models. Although the parameterization we described are for spherical HLPs, they can be easily adapted to other shapes by considering their respective geometric properties.

Our on-the-fly stylization and hatching techniques also opens interesting possibilities and can also be used for other objects than plants with minimal changes. Compared to traditional, object-space parameterizations, they are easy to compute in real time, which makes them an attractive alternative for interactive applications. On the other hand, as they are based on approximations, coherence problems can occur: hatch and silhouette texture sliding and discontinuities. The artifacts are less noticeable for the complex structures of plants which tend to hide them. We hope to find further improvements in this direction by refining the HLP re-modeling process.

Acknowledgement

We would like to thank the German Environment Foundation (DBU) for supporting this work within a PhD grant.

References

BEHRENDT, S., COLDITZ, C., FRANZKE, O., KOPF, J., AND DEUSSEN, O. 2005. Realistic real-time rendering of landscapes using billboard clouds. In *Eurographics 2005 Conf. Proc.*, ACM SIGGRAPH.

CURTIS, C. J. 1998. Loose and sketchy animation. In *SIGGRAPH '98: ACM SIGGRAPH 98 Electronic art and animation catalog*, ACM Press, New York, NY, USA, 145.

DEUSSEN, O., AND STROTHOTTE, T. 2000. Computer-generated pen-and-ink illustration of trees. *Computer Graphics, SIGGRAPH 2000 Conf. Proc. 34*, 4, 13–18.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Interactive visualization of complex plant ecosystems. In *IEEE Visualization 2002*, IEEE, 219–226.

GOOCH, A., GOOCH, B., SHIRLEY, P., AND COHEN, E. 1998. A non-photorealistic lighting model for automatic technical illustration. *Computer Graphics 32*, Annual Conference Series, 447–452.

GOOCH, B., SLOAN, P.-P. J., GOOCH, A., SHIRLEY, P., AND RIESENFELD, R. 1999. Interactive technical illustration. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 31–38.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, K. Akeley, Ed., 517–526.

KALNINS, R. D., DAVIDSON, P. L., MARKOSIAN, L., AND FINKELSTEIN, A. 2003. Coherent stylized silhouettes. *ACM Transactions on Graphics 22*, 3 (July), 856–861.

KOWALSKI, M. A., MARKOSIAN, L., NORTHRUP, J. D., BOURDEV, L., BARZEL, R., HOLDEN, L. S., AND HUGHES, J. F. 1999. Art-based rendering of fur, grass, and trees. In *SIGGRAPH '99*, ACM Press/Addison-Wesley Publishing Co., 433–438.

LAKE, A., MARSHALL, C., HARRIS, M., AND BLACKSTEIN, M. 2000. Stylized rendering techniques for scalable real-time 3d animation. In *NPAR 2000 : First International Symposium on Non Photorealistic Animation and Rendering*, 13–20.

MARKOSIAN, L., KOWALSKI, M. A., TRYCHIN, S. J., BOURDEV, L. D., GOLDSTEIN, D., AND HUGHES, J. F. 1997. Real-time nonphotorealistic rendering. *Computer Graphics 31*, Annual Conference Series, 415–420.

NORTHRUP, J. D., AND MARKOSIAN, L. 2000. Artistic silhouettes: a hybrid approach. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, ACM Press, New York, NY, USA, 31–37.

PRAUN, E., HOPPE, H., WEBB, M., AND FINKELSTEIN, A. 2001. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 579–584.

RASKAR, R., AND COHEN, M. 1999. Image precision silhouette edges. In *1999 ACM Symp. on Interactive 3D Graphics*, ACM SIGGRAPH, 135–140.

REEVES, W., AND BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *Computer Graphics (SIGGRAPH '85 Conf. Proc.)*, vol. 19, 313–322.

SAITO, T., AND TAKAHASHI, T. 1990. Comprehensive rendering of 3-d shapes. *Computer Graphics, SIGGRAPH 90 Conf. Proc. 24(4)*, 197–206.

SALISBURY, M. P., ANDERSON, S. E., BARZEL, R., AND SALESIN, D. H. 1994. Interactive pen-and-ink illustration. In *SIGGRAPH '94*, ACM Press, 101–108.

SALISBURY, M., WONG, M., HUGHES, J. F., AND SALESIN, D. 1997. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH 97 Conf. Proc.*

SASADA, T. 1987. Drawing natural scenery by computer graphics. *Computer-Aided Design 19*, 4, 212–218.

SMITH, A. 1984. Plants, fractals and formal languages. *Computer Graphics (SIGGRAPH 84 Conf. Proc.) 18*, 3, 1–10.

SOUSA, M. C., AND PRUSINKIEWICZ, P. 2003. A few good lines: Suggestive drawing of 3d models. *Computer Graphics Forum 22*, 3 (Sept.), 381–390.

WILSON, B., AND MA, K.-L. 2004. Rendering complexity in computer-generated pen-and-ink illustrations. In *NPAR 2004*, 103–112.

WINKENBACH, G., AND SALESIN, D. 1996. Rendering parametric surfaces in pen and ink. *Computer Graphics, SIGGRAPH 96 Conf. Proc. 30*, 4, 469–476.

XU, H., AND CHEN, B. 2004. Stylized rendering of 3d scanned real world environments. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, ACM Press, New York, NY, USA, 25–34.

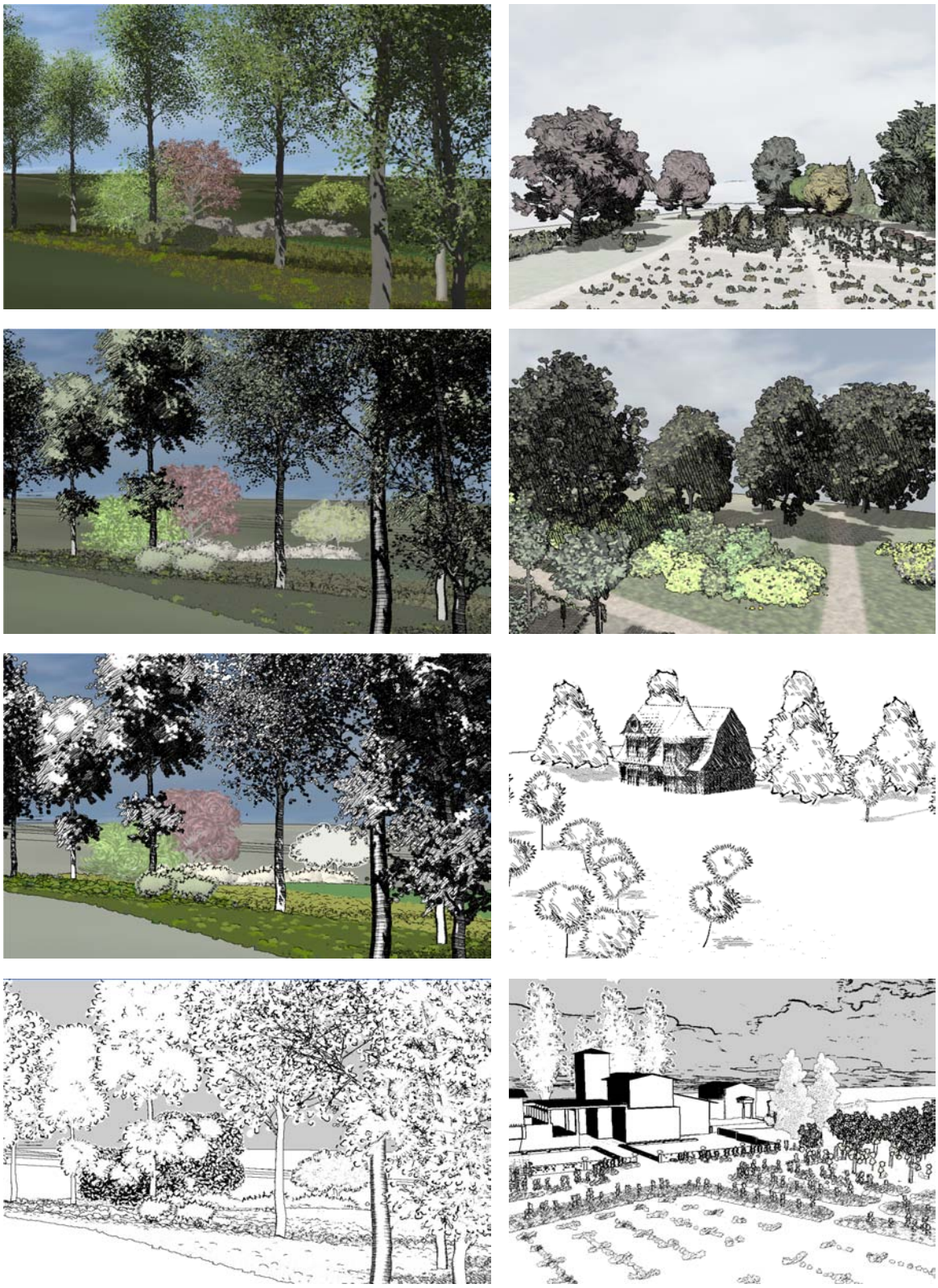


Figure 3: Left column, from top to bottom, continued abstraction of a landscape: photorealism, half-NPR, full NPR, silhouettes and hatching only. Right column: several other abstract renditions, from top to bottom: Van Gogh look (abstract, elliptical leafs around the HLPs), pointilistic style, minimalistic drawing, combination of different styles.