GPU Real-Time Medium Model for Artistic Temporally Coherent Rendering

Belhadj Farès, Suarez Jordane, Boyer Vincent
L.I.A.S.D
Univerité Paris 8
Saint Denis, France
Email: amsi, suarez, boyer@ai.univ-paris8.fr

***Abstract*—We present a consistent model for artistic medium reproduction in 3D scene renderings and animations. Computer generated stylized animations should reproduce the medium used by the artists while avoiding the artifacts present in hand-drawn animations. In our model, for each scene object, we create a surface grain according to the object geometry and apply it as additional material properties. Our system aims to make these additional material properties remain consistent to the scene / objects animation while avoiding the *shower door* and popping effects. Moreover, in contrast with existing methods in which coherence is primarily maintained during walkthroughs, our strategy consists in maintaining the grain attachment for any object rotations and non z-axis (i.e. depth) displacements and refining the grain density according to z-axis displacements. Furthermore to prevent this artifact, we propose to select salient additional material properties and reconstruct it for the whole image through a fractal surface reconstruction process.**

Our model is fully implemented on GPU, results are obtained without any adaptation of the original 3D scene or objects and the rendering process is real-time.

I. Introduction

Creating animations using traditional medium by hand is a tricky task; artists paint or draw each frame independently and could not ensure a frame to frame coherence. Aleksandr Petrov is well-known for these Paint-on glass animations (see for example "The Old Man and the Sea" [1]) using different painted glass sheets to represent the multiple levels of each frame. After a snapshot, the painting is slightly modified for the next frame. This process is very expensive (i.e. it took over two years for the 29000 frames of "The Old Man and the Sea") specially while avoiding undesirable effects that can appear and disturb the viewer. Commonly the animators fill in their drawings with flat color paint to ignore these constraints.

This challenge is very interesting in computer graphics and particularly in non-photorealistic rendering domain since proposed solutions assist the user in complex artistic tasks. To obtain a successful temporal coherent stylization, proposed methods should take into account the frame to frame coherence to avoid undesirable effects such as popping. Note that achieving a complete temporal coherence is not strictly a requirement, controlled variations (i.e. having a behavior that seems to follow the animation flow) can provide additional benefits to the produced animation (for example the sea rendering in Petrov's animation).

Many authors have defined goals that should be achieved for successful temporally coherent stylizations. Litwinowicz [2] has mentioned, in the discussion section of his paper, some of the undesirable effects produced by its impressionist stylization system for video. More largely, Hertzmann [3] has provided considerations on how do artists create imagery, how do observers respond to artistic imagery and how do we evaluate aesthetics. He has noticed that measuring subjective qualities of art is a tricky business and has mentioned two main metric categories: "proxy metrics" that does test measurable properties relevant to evaluating art (biophysical measurements, fractal dimension statistics or improvement of the memory performance) and "asking artists" that consists in informal feedback from artists. Recently, Bénard et al. [4] have identified three goals to ensure a temporal coherence: the flatness, the coherent motion and the temporal continuity. The flatness is the ability to convey the 2D aspect of a given stylization. The coherent motion provides a high correlation between 3D scene and pattern motions and consequently avoids the *shower door* effect. The temporal continuity consists in guaranteeing the frame to frame coherence. They have summarized the trade-offs made by various solutions for the temporal coherency problem. They have noted that these have a conflicting nature and any solution is necessarily a compromise.

We share the same idea, any such method necessary incur some artifacts. Moreover we think that each method has its own advantages and drawbacks in term of usage and variety of produced effects, and nobody can predict what kind of effects the artists want to produce: does an impressionist painter draw in 2D (as constrained by the canvas) the result that he designs in 3D and what would he do if he would animate the whole?

The main contribution of the paper consists in producing a coherent stylization to maintain grain attachment on each object. We create a surface grain as additional material properties for each scene object according to its geometry. These material properties are used to produce a coherent stylization. As previously mentioned, our solution is a compromise and we favor the flatness and temporal continuity. So these additional material properties remain consistent according to the scene / objects animation while avoiding *shower door* and popping effects. While existing methods try to primarily maintain coherence during walkthroughs, we priorly maintain the grain attachment (i.e the aspect / density) for any object / viewer rotations and non z-axis (i.e. depth) displacements and refine the grain density according to the z-axis displacements. In fact,

variations could be perceived only through z-axis displacements. Moreover by opposition to the existing methods in which the noise is computed and then applied to the object we propose an inverse approach: according to the object coordinates (position, normal, texture) we compute the noise. To prevent the z-axis motion coherence artifact we use an additional fractal surface reconstruction model. This model is able to extract salient characteristics and to reconstruct the whole image. The reconstruction process combined to the medium generation offer a perfect motion coherence and preserve flatness. Temporal continuity is almost preserved.

Note that as additional result, this permits to obtain traditional medium such as impressionism through visible brush strokes effects. All of these contributions have been fully implemented on GPU and the rendering process is real-time.

In the following, we first present the related work, then we focus on our model and particularly on the grain surface generation and the additional noise generation through the fractal surface reconstruction. After, a framework is proposed to produce artistic medium. Finally, the results are presented and well discussed.

II. Background

Reproducing artistic medium model such as painting, pen-and-ink is one of the main field of the non-photorealistic rendering domain and a large collection of models have been proposed in the last two decades [5], [6]. Their application for the generation of stylized animations is not obvious since temporal incoherence introduces visual artifacts which are unsuitable for the audience. Even if many methods try to address the temporal coherence for stylized animations, the conflicting nature of flatness, motion coherence and temporal continuity could only provided compromise solutions [7].

Techniques for rendering animations in a painterly style have to solve two main problems: produced images have to be coherent over time; medium application should stick to the animated surfaces to avoid the *shower door* effect.

Meier [8] proposed the first method for rendering animations in a painterly style using particles. These are created to represent the geometry and, according to the distance from the viewpoint brush strokes, used to paint the model. Unfortunately the distribution is realized in the object-space and can be non-uniform (too dense or too sparse) after its projection.

Chi et al. [9] recently presented a framework for interactive 3D painterly rendering. The 3D model is described through a multi-resolution bounding sphere hierarchy which is computed according to color and geometry information. This representation is then used to select candidate regions and stroke nodes in order to apply stroke abstractions. View dependent or independent stroke abstractions are proposed but *shower door* and popping effects respectively appear. Note that smooth level transitions are introduced to reduce the popping effect. The main contribution of this paper is a hierarchical representation of 3D objects used to efficiently abstract painterly rendering but the animation problems remain.

To improve the feeling of 3D motion, Cunzi et al. [10] have introduced the notion of dynamic canvas. Canvas is dynamically animated according to the camera movements reducing the *shower door* effect for certain transformations. A solution based on 2D transformations and a spherical distortion is used to describe 3D motion in animated scenes and also to animate background canvas. This approach is restricted to a set of camera motions and particularly designed for walkthrough.

Kaplan et al. [11] have proposed a generative model for dynamic canvas motion. In this model papers are created through many fibers in which each of them has a position, direction and width; impurities can also be added into the paper model. To provide dynamic canvas, individuals fibers are associated with each scene object including the background. A background is modeled using a triangular mesh and always sits at a constant distance and orientation from the viewpoint. This background and objects are then dissociated avoiding the *shower door* effect. Unfortunately, this model is hardly limited to canvas fibers, is not real time, and suffers from popping effects.

Bousseau et al. [12] have presented a pipeline to obtain watercolor images. Starting with an image or a 3D model, they first produce abstracted image through abstraction steps and then apply watercolor effects. Watercolor effects consist in successive treatments including paper effects; edge darkening and pigment density variation. They have proved that watercolor can be simulated using low frequency turbulent flow and high frequency pigment dispersion. Two dynamic canvas approaches have been proposed to avoid temporal coherence problems. Unfortunately, this texture attachment method is only well adapted for a single object viewed with a static background and flow textures method needs, for every frame, the interactive generation of noise and turbulent textures. In addition to time consumption, the popping effects are not totally avoided with these methods.

A solution to provide temporal coherence for video water-colorization was also proposed by Bousseau et al. [13]. It combines texture advection presented above according to the optical flow field computed for the video and mathematical morphology operation to produce a temporally coherent abstraction of the video.

Vanderhaeghe et al. [14] focused on temporal coherence of stroke based renderings. The method is based on a point distribution in the image plane. These points are projected into the underlying scene to apply the scene motion. At each frame, points can be added or removed if needed. As mentioned by the authors, their image-based distribution approach is well-adapted for a statistical distribution measures but an object-based approach [8] gives much better animation results. In fact, their distribution is more coherent than static ones but mechanisms like blending and sliding are proposed to ensure temporal coherence.

Recently, Benard et al. [4] have proposed a stylization based on the Gabor noise function. The noise primitive

is used as a procedural texturing to create temporally coherent in 3D animations. It is defined in 2D screen space and a point distribution on the surface of the 3D model is used to ensure the coherence of motion. Due to the 2D definition and the point distribution, popping effect appears. Moreover, time computation for this method does not provide real-time rendering for complex 3D scenes (from 50 fps to 8 fps for very small scenes composed with 4k to 50k triangles).

Finally Kass et al. [15] have proposed a solution based on a Perlin noise filtered by the depth and the velocity fields and their consequent occlusion relationships. Stylizations are also possible applying the coherent noise on image dilatation/erosion/warp for example.

Note that all of the coherent noise approaches provide a solution from noise space to object space. With this kind of approaches, the noise is coherent on the resulting image if stationary noise (independent on the frame i) is provided and disocclusions (surface point has just become disoccluded) are treated. Due to these constraints, these methods are sensitive to motion that causes extremely rapid changes.

III. Granularity, from object space to image space

In our solution, we provide a coherent noise. As Kass et al. [15], our solution is based on a Perlin noise. But by opposition to the previous noise approaches that first consider a noise and then its application on the 3D geometry, we propose an inverse approach: starting with the 3D geometry, we provide a function which is able to compute consistent noise. The main advantage of this approach resides in the fact that no problem remains when geometry appears; disocclusion is not a particular case and rapid changes is no more a problem to solve.

So to produce medium, we propose to generate a surface grain that is specific to each scene object while maintaining a uniform frame overall. Consequently we provide a model able to:

- compute the surface grain of each object independently. The grain surface can be defined as additional material properties. These are attached to the object and our model guaranties the continuity of the grain surface (the noise construction is consistent over movements);
- guaranty a grain surface continuity around the scene. Even if each object has its own grain surface, viewer does not perceive any discontinuity on the scene.

These new material properties, representing the grain surface, would be generated using two types of data:

- a procedural noise function used to simulate the medium effect;
- one (or a blend of) texture given as an input of the pipeline (see section 5) and used to reproduce canvas.

Thus, the question would arise from the calculation of correct settings that respectively fit the following constraints for each data type:

- getting the appropriate parameters (one, two, three, ..., n) for the chosen noise function;
- getting the texture coordinates.

Remind that the grain must remain a material property that is associated to objects and follow them as best as possible.

Like Bousseau et al. [12] and Meier [8], we want that the produced effect sticks to the object surfaces during animation in order to avoid the *shower door* effect; here, two types of movements are distinguished: depth variations, for example walkthroughs, and other types of transformations. In contrast with existing methods where the main idea is to maintain a coherent and uniform canvas during walkthroughs, we choose to give a full priority to the grain attachment during object rotations and other non z-axis (i.e. depth) displacements. On the other side and in terms of grain density, we maintain uniformity of the overall frame according to displacements along the depth-axis. Thus, this leads us to try to compute the grain as a function related to a 3D coordinate that is strongly object-space dependent while taking into account the depth of the projective space.

A. Noise parameters

We want to generate data in order to simulate a medium close to those used in artistic medium such as oil painting, pencil, etc. Noise functions are frequently used to generate this type of data. Also, the procedural aspect due to its local construction is interesting since it helps to keep control over the result continuity in the definition space. Perlin's noise [16], [17] is probably the most famous and widespread procedural noise function. It covers a large part of our needs and is one possible option to generate our medium. The constraints introduced above are expressed in terms of new materials that are fixed to the target 3D model. Thus, we need to calculate a 3D coordinate as a parameter of the Perlin's noise function. This 3D coordinate depends on the vertex coordinate $\overrightarrow{\mathcal{V}}$, (to obtain a consistent grain surface on the movement) and the projective space, $\mathcal{P}$ (to have a grain continuity around the surface scene) and can be formulated as:

$$\mathcal{T} = \begin{bmatrix} x_t \\ y_t \\ z_t \\ w_t \end{bmatrix} = \mathcal{P} \times \begin{bmatrix} x_v \\ y_v \\ z_e \\ 1 \end{bmatrix}$$

where $\overrightarrow{\mathcal{V}} = (x_v, y_v, z_v, 1)^t$ is the vertex homogeneous coordinates in the object space; z_e the z-axis component of $\overrightarrow{\mathcal{V}_e} = \mathcal{M} \times \mathcal{V}$ which expresses the vertex depth in the eye space ($\mathcal{M}$ the model-view matrix); $\mathcal{P}$ the projective space matrix.

As argued above, when representing objects in the eye space expressed by a model-view matrix $\mathcal{M}$, only the depth component will be altered by using this matrix. Thus, for a vertex having a homogeneous coordinates in the object space, we keep x_v, y_v and compute its z_e. Then, we obtain the vector $\mathcal{T}$ which expresses this amount that

varies with depth (z) and remains constant for other types of transformations (x and y) in projective space.

Note that for what follows $\mathcal{T} = (\frac{x_t}{w_t}, \frac{y_t}{w_t}, \frac{z_t}{w_t}, 1)^t$.

As an example, figure 2 (a) presents a 3D model and from image (b) to (e) the grain surface provided when $\mathcal{T}$, computed in the vertex stage, is used as parameter of the simplex noise function that returns the fragment color at the fragment stage. One can remark that the produced noise on the background is constant. On the images (c) and (d), as the noise computation depends on the vertex coordinates, we can notice that the noise is anchored to the object. On the image (e), one can remark that the frequency of the noise has changed on the object.

As a conclusion of this part, in term of procedural noise function, the noise parameters anchors a consistent noise to the object and adapts the noise frequency according to the depth variations.

B. Texture coordinates

Figure 1. Projecting 3D texture coordinates on cube faces in order to obtain multiple 2D coordinates: red, yellow and green triangles have all their vertices respectively projected on YZ, XZ and XY planes. Blue triangles have vertices which are projected on different planes.

As our grain surface is represented both by a procedural noise function and a texture, it is obvious that properties of the procedural noise function should be adapted to the texture. In term of texture, we should both provide a consistent texture coordinates for each vertex to anchor the texture to the object; adapt the texture coordinates for each vertex according to the depth to provide the adapted frequency.

The vector $\mathcal{T}$, used as a noise parameter, is also used to compute the texture coordinates. As we use 2D textures, we provide a consistent transformation to compute, for each vertex, a 2D texture coordinate using $\mathcal{T}$. We propose to use a method inspired by environment mapping techniques to compute this transformation. Here we project 3D vertices in order to get 2D coordinates. By analogy with environment cube mapping, we project the vertex normal $\overrightarrow{\mathcal{N}}$ coordinates onto one of the six cube faces and store the result in $\overrightarrow{\mathcal{C}} = (x_c, y_c, z_c, 1)^t$. This projection guaranties that one of these coordinates always corresponds to one of the six planes: x_c (respectively y_c, z_c) if the projection of $\overrightarrow{\mathcal{N}}$ is on left or right (respectively top/bottom, front/back) cube face. We use this coordinate to select which coordinate is not used in the vector $\mathcal{T}$. In practice, the normal projection on a cube mapping is used to select two of the three coordinates of $\mathcal{T}$ hereafter used as vertex texture coordinates.

Note that, for a given geometry primitive (a triangle), we can not ensure that every normal projections are on the same cube face. The blue triangles in figure 1 illustrate this situation. In other words, for a given triangle, we have to compose with multiple textures. In that particular case, we create multiple texture coordinates (one per plane) in the geometry shader. Then for each fragment we mix the considered textures.

As an example, figure 2 from image (f) to (i) presents the texture mapping corresponding to images from (b) to (e). Figure 2 (j) presents a complete granularity result combining noise and texture techniques (i.e. it is not simply a combination of two result images).

Note that we provide consistent and coherent texture coordinates with this method, because: for a given vertex where its normal vector is expressed in the object space, the two selected coordinates of $\mathcal{T}$ are always the same; the value of theses two selected coordinates of $\mathcal{T}$ depends on the projection including the depth of the object in the viewer space.

In conclusion of this part, in term of texture coordinates generation and texture mapping, the noise parameters anchors consistent texture coordinates to the object and adapts the texture repetition according to the depth.

In general conclusion, both in term of procedural noise function and in term of texture coordinates generation and texture mapping, the noise parameters anchors a consistent noise to the object and adapts the noise frequency according to the depth. As illustrated hereafter in the results and in the additional video, we provide a solution for all kinds of object (deformable objects, fluids, complex scene...).

Note also that the object space infinite zoom mechanism proposed by Cunzi et al. [10] and extended by Bénard et al. [18] is not adapted to our medium generation since we use a Perlin noise generation.

IV. More noise: Fractal Surface Reconstruction

As mentioned above, we choose to give a full priority to the grain attachment during object rotations and other non z-axis (i.e. depth) displacements. Thus, motion coherence artifacts on the z-axis can be visible with the additional material generation model. To prevent this artifact, we propose to select salient features and reconstruct it for the whole image. Two problems remain: how to find salient informations and how to realize the reconstruction?

In [19], the authors have suggested that various painting effects can be realized with a fractal reconstruction model applied to characteristics expressed as heightmap data. It has been demonstrated that their reconstruction model preserves the given characteristics. The fractal reconstruction model have been first presented and detailed in [20]. Their model provides an efficient solution to produce impressionist effects and compared to other reconstruction methods (ex. see comparison given in [20] with the RBF method [21]) the time computation is quite reasonable (i.e. about 2 fps) for an interactive reconstruction.

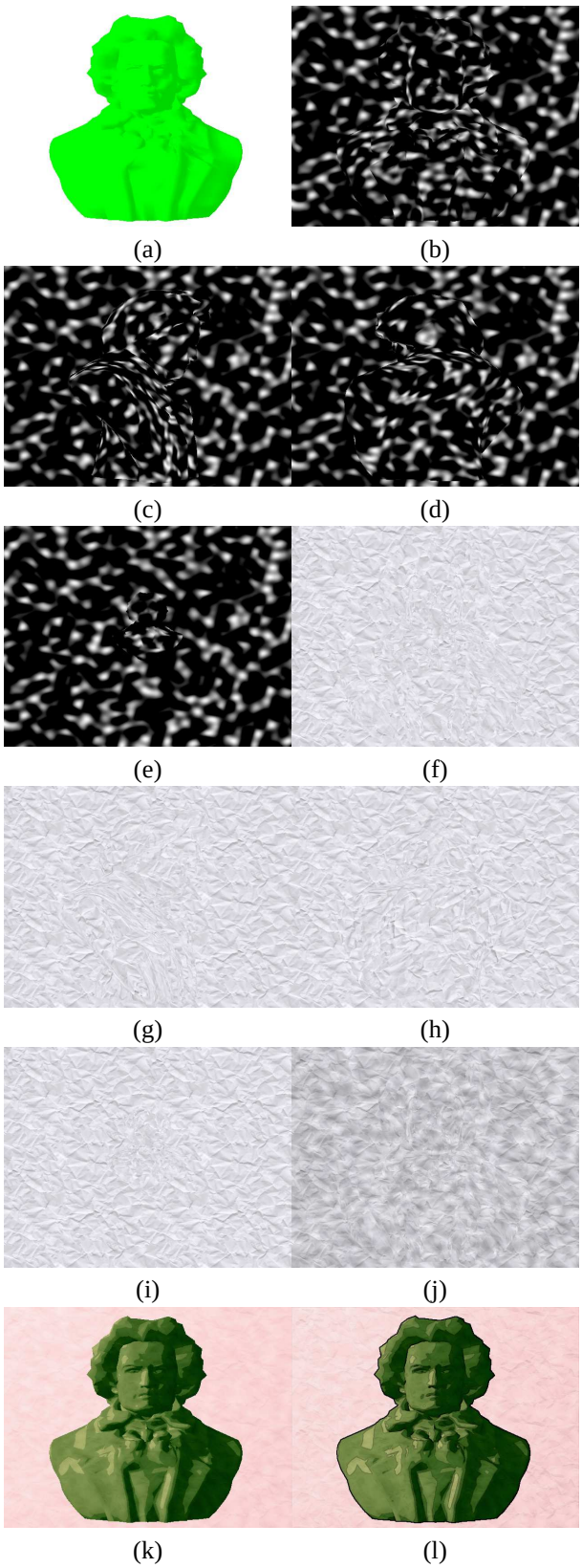

Figure 2. Grain surface generation through noise and texture techniques: (a) original 3D model; from (b) to (e) generation of grain surface and its application on the model; from (f) to (i) texture mapping corresponding to grain generation from (b) to (e); applying result on the model using lighting (k) and Sobel filter (l).

We propose to adapt this fractal reconstruction algorithm to stylized animations. Thus, we have first to express the image characteristics at each frame in order to improve the motion coherence. In our work the reconstruction process should be largely improved since real-time animation should be provided. In the following we treat these issues. Note that depending on the level of abstraction, the reconstruction result will differ from the original frame in inverse proportions.

A. Extraction

Here, the results produced in the previous section is used as the initial data. We propose to extract a set of various characteristics (different abstraction steps, data for deferred rendering) using them as possible constraints of the reconstruction process. These should preserve the surface grain previously generated. First, we use a threshold function on the generated medium. As an implementation detail, we precise that the medium is stored on the alpha-channel of the initial data. Consequently the surface grain appears as one of the most preserved characteristics. The figure 3 presents in the first row the output of the additional material generation output. Then we provide, on the second row on the left, the result of the extraction process considering a threshold on the luminance of the medium.

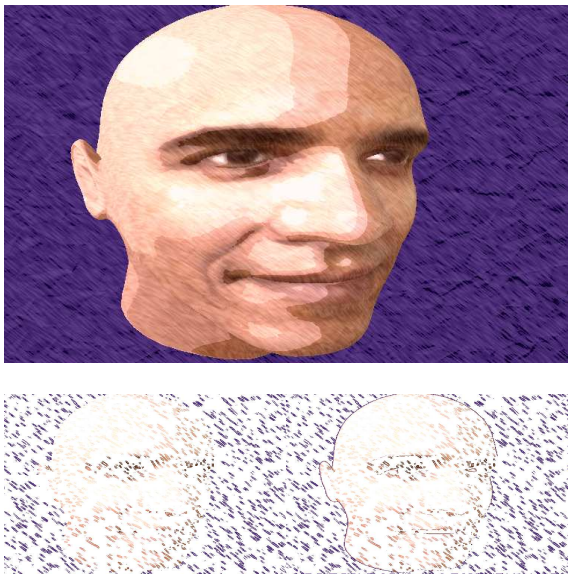

Figure 3. Extraction step. From top to bottom and left to right: result of the additional material step; extraction obtained using threshold on the medium component; extraction obtained using threshold on the medium component and a Sobel edge detection.

More than surface grain, the contours of the scene objects can also be considered as an important characteristic. If needed by the user, a Sobel edge detection can be computed using the image luminance and a threshold on the result used to select contours extracted for the reconstruction process. The image on the right of the second row shows the case where medium and contours characteristics are extracted.

B. Reconstruction

The Morphologically Constrained Midpoint Displacement method (MCMD) [20] is mainly a surface interpolation process which is able to take into account constrained elevations while using random displacements that depend on the subdivision level. Using this method we can modify the random variations in order to control the fractal dimension of the produced image. It is based on midpoint displacement methods like the diamond-square one [22]. This family of iterative subdivision methods can be resumed to a tree traversal process that looks to parent node values to compute children ones (i.e. a top-bottom process). In the MCMD method, the first sub-process (bottom-up) serves to take into account constraints expressed at any tree position (i.e. root or any children nodes) by backing up the information to ancestors. Parent values are extrapolated from their children ones. Unfortunately as previously mentioned, none of them [19], [20] have proposed a real-time implementation.

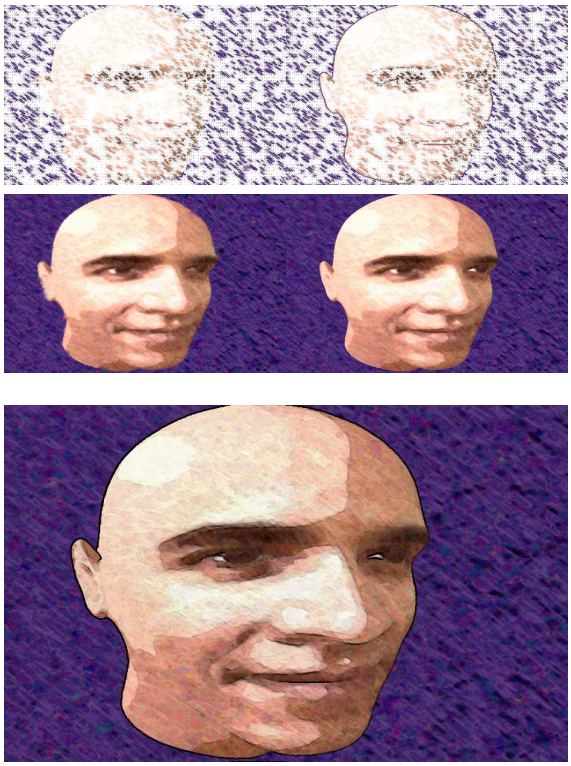

Figure 4. Different steps of the fractal reconstruction process. From top to bottom and left to right: the extraction (without or with contour) is considered as given constraints for the MCMD method and constraints are forwarded to their ancestors in a bottom-up process; The reconstruction is achieved in a top-bottom process (considering the result of previous row). As a post-processing, we add contours on the previous result (second row on the right).

In this work, we do not aim the same goal as [20] in which parent constraints (i.e. elevations) are computed according to the children ones. From a parent node point of view, we just need to know that a constraint exists for a child and thus the parent should consider its color in the original image as a constraint (i.e. we do not have to extrapolate its value, we just read it from the input). So the

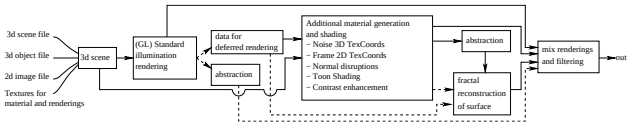

Figure 5. Artistic rendering pipeline including medium surface generation and fractal reconstruction.

algorithm consists in looking for a given fragment where the color is flagged to unknown if one of its children has a color (i.e. has a constraint / is known).

In that case, the fragment updates its value using the pixel color in the original image (i.e non-abstracted). This fragment program should be repeated n times where n is the tree depth (i.e. $n = \lceil ln_2(max(W, H)) \rceil$, where $W \times H$ is the image resolution). As an implementation detail, we precise that the entire algorithm tree is stored as a texture. Thus for each fragment, we can quickly access to its children coordinates through this texture. The top row of the figure 4 shows the obtained result after this first sub-process.

For the second sub-process, it can be resumed to midpoint displacements applied on each color channel (R, G, B) where some of the subtrees are (now) constrained with values that are already known. Here again, a fragment program is repeated n times simulating a top-bottom subdivision process. In this process and for each coordinate, we should determine when the coordinate value must be computed regarding to the subdivision order. We pre-compute the computation level for each fragment position and store the result in a texture. Furthermore, we need to know the list of each child node parents in order to compute its value if necessary. Then we also choose to pre-compute these data and store them in an other texture. Thus, at each subdivision level given as a fragment program uniform parameter, we can determine if the fragment value should be computed (i.e. has the right level). For those with the right level, we test if the value is unknown and in that case interpolate it (with or without displacement) using the fragment parent values. The middle row of figure 4 shows the final result obtained after the midpoint-displacement sub-process. Note that the result differs between the two reconstructions. On the left image, as we do not include the contour as a characteristic during the extraction step, the reconstruction process spreads information without any consideration on the objects. This produces visible brush strokes that exceed objects borders. On the contrary, on the right image, one can see that constraints have been forwarded to the reconstruction process and the visible brush strokes are bounded to the objects borders.

Considering the adaptation of the first sub-process and a full GPU implementation through texture generation, we obtain a real-time reconstruction and our new algorithm is 35 times faster than the one of Belhadj et al. [20].

V. Framework

We propose an artistic rendering style process (see figure 5) that considers additional object properties and uses

noise to produce images and temporal coherent animations of 3D scenes.

The main part of our pipeline is *the additional material generation and shading* and *fractal reconstruction* described in the previous sections. During the same rendering pass, we are able:

- to disturb the normal of each fragment depending on the generated noise. We disturb each normal component with $\mathcal{T}$ in order to obtain a 3D noise. Note that we can modify $\mathcal{T}$ (scale and rotation) before this perturbation stage providing a collection of various results presented in the results section. On the figure 6, the normal disruption (higher disruption) is illustrated on the left image of the fourth line.
- to disturb the lighting of each fragment depending on the texture generation. In that case, we compute the luminance for each fragment (given by the texture and the color material). This luminance can be used hereafter.
- to create a toon shading. Since it is a well-known effect, one can see that it is used on every images of the figure 6. Note that the normal perturbation presented above is then combined with the fragment normal to compute the fragment color using a toon shading. Moreover the luminance can weight the scalar product used as the toon shading 1D texture coordinate (see the last image on figure 6).
- to enhance the contrast of each fragment. This is obtained by applying a grain extraction between the image and its inverse luminance layer. In practice, we achieve this operation at the end of the fragment shader.

The other parts are used as a pre-computation (left part of the figure) or as a possible post-treatment (right part of the figure) and demonstrate the granularity integration in an artistic medium rendering pipeline.

Note that the pre-treatment is only designed to extract data for post-treatment. Even if this step is represented on the left of the additional material generation step of the figure, they are independent and both can be realized at the same time. We realize a "standard illumination rendering" using all parameters given by the user (camera position and orientation, lighting and textures): to extract data for deferred rendering (ex. normal map), to be potentially used in a fractal reconstruction of surface; to create abstraction images of the 3D scene. The abstraction images consist in creating images like contours and segmentation (through a lighting model), and can be mixed at the last step to produce the final image.

Finally the produced image is generated by potentially mixing pre-treatment rendering (abstraction and standard illumination rendering), additional material generation and shadings and fractal reconstruction. Depending on the user choice, some of them can be used or ignored.

In the figure 5, the solid lines represent rendering data transfers and dashed ones depict any other data transfers.

VI. Results and Discussion

We present a collection of results obtained with our framework and a discussion concerning the temporal coherence evaluation.

As one can see, we present results produced both on 3D scenes and on 2D images. Figure 6 presents an overview of possible results produced by our model. We first create the noise, adapt its frequency, orientation and then apply it to the model. On the bottom, we add contours, use the normal perturbation and provide fractal reconstruction. Even if the noise simulates the support of an artistic medium such as canvas, the normal perturbation, considered as another additional material, provides additional effects such as painting volume.

We also apply a our model to a complex scene including a landscape, houses, and an airship (see figure 10 and 11). As one can see, our method is robust and does not suffer to any problem due to the complexity of the scene. Consistent noise anchors to each object and flatness is preserved on the entire scene.

These figures illustrate one of the advantage of our approach: according to the model geometry, we compute parameters that permit to apply noise avoiding the *shower door* effect. We provide various results to illustrate painting and pencil reproduction.

Note that we obtain at least 100 fps on $\approx$ 1M polygons scene rendered at a 756×480 resolution on an Intel Bi-Xeon 2.7 Ghz and an nVidia Quadro FX 3800 ($\approx$ 35 fps when the scene is rendered at a 1600×1200 resolution). When using our reconstruction, in the most unfavorable case, we obtain at least 30 frames per second with the fractal reconstruction surface.

Video submitted as additional material shows that temporal coherence is provided during displacements (video is also available at http://www.ai.univ-paris8.fr/~amsi/papers/sitis2012/). A *resampling effect* only appears for very high depth variations between two consecutive frames. Using, the surface fractal reconstruction process, this effect tends to disappears.

As mentioned by Bénard et al. [7], no such ground truth exists for non-photorealistic rendering techniques especially in animations. Generally a visual inspection is realized by the authors. At the most, some criteria are evaluated by a set of subjects and an aesthetic overall on the sequence is given. As one can see, there is no formal and perceptually grounded evaluation and even in the state of the art on this topics [7], the authors just imagine objective measures. In practice, as described above and as mentioned by most of authors, a stylized animation method pursues contradictory goals: flatness, coherence motion and temporal continuity. Thus a comparison based on ranking on images or video sequences for each criteria is the adopted solution. We choose the same presentation as the one proposed by Bénard et al. [7] and add more details.

Table I summaries the trade-offs made by our different rendering techniques both in the best and worst case.

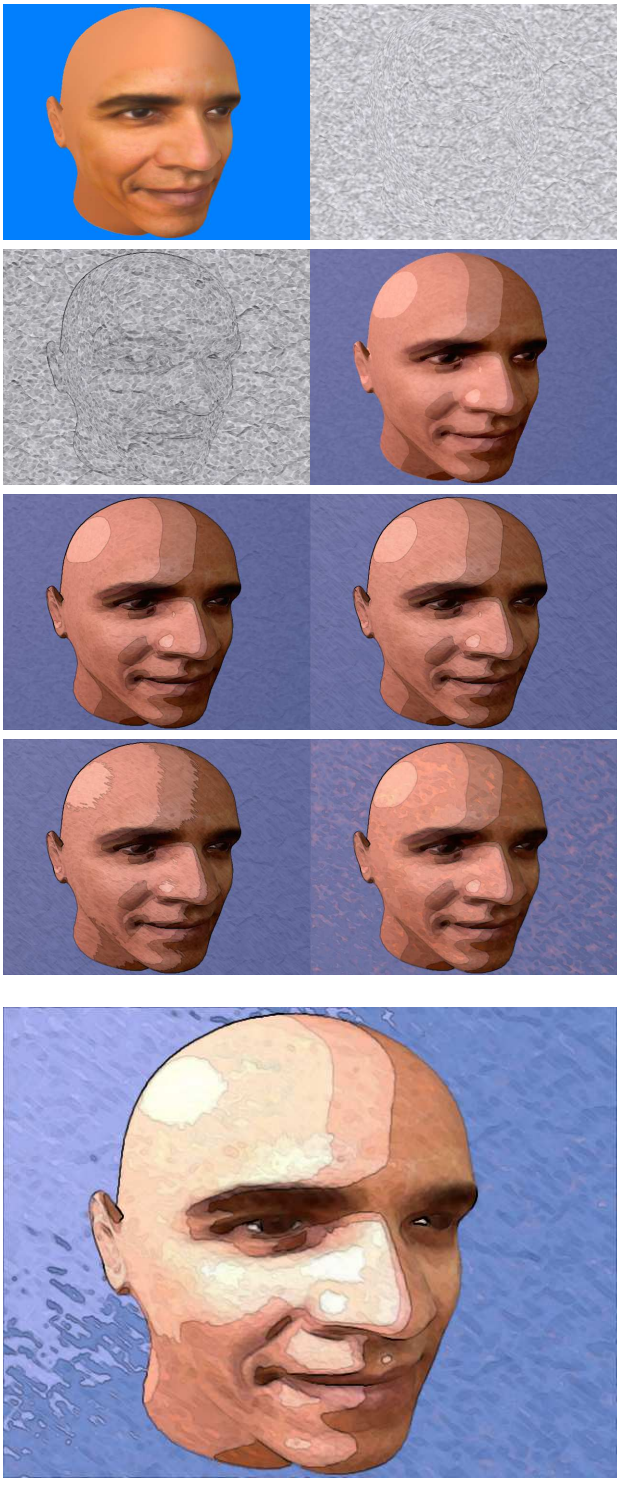

Figure 6. Top line: the original model; noise application on the model. Second line: adding contours; application of additional noise material and toon shading. Third line: noise variations with contours and toon shading. Fourth line: adding normal perturbation; using MCMD reconstruction. Last line: the result is produced using noise additional material, normal perturbation and median filter.

For the addition material generation (noise generation and attachment):

- flatness is almost reached. The geometry is slightly perceptible due to the 2D texture cube mapping (blue triangles on figure 1);
- motion coherence is very good excepted for z-axis displacements. As previously described, we give a full priority to the grain attachment during object rotations and other non z-axis displacements. So artifacts are visible during walkthroughs and for objects with a high variations on z-axis. This effect can be slightly attenuated by a multiple scaled noise;
- temporal continuity is almost preserved. In theory temporal continuity depends on the noise frequency chosen by the user. In the worst case, when the noise frequency is too high regarding the windows resolution, abrupt changes can appear from frame to frame. Note that this effect is produced only according to the user choice, it is not an artifact produced by our method.

	Flatness		**Coherent motion**		**Temporal continuity**	
	Worst	Best	Worst	Best	Worst	Best
Material Generation	+	++	+	++	+	++
Fractal Surface Reconstruction	++	++	++	++	-/+	+

Table I
SUMMARY OF THE TRADE-OFFS MADE BY OUR MODEL

The figure 7 presents an average of these results through a diagram where each goal is represented by an axis.

For the addition material generation with the fractal surface reconstruction, we improve the motion coherence to the detriment of temporal continuity. Flatness is better preserved but its aspect change due to the reconstruction process. Also, the reconstruction process combined to the medium generation offers a perfect motion coherence. But in that case, the temporal continuity is disturbed. Indeed the reconstruction process can not guarantee a perfect frame to frame coherence.

The figure 8 presents the average of these results through a diagram where each goal is represented by an axis.

As a complementary result, we want to focus on a particular painting style, the impressionism. As mentioned in [23] it is one of the most accomplished technique of painting where realistic scenes are realized. The impressionist painting is mainly characterized by visible brushstrokes, open composition, emphasis on light and color arrangement. Short brush strokes of pure and unmixed color are used in order to achieve the effect of intense color vibration. For an impressionist painter the optical mixture in the eye replaces mixture of the pigments in the pallet. The figure 9 presents four paintings. Each of them respects the criteria described above but some of them have been generated through our system. Enjoy these and find which one(s) is (are) not exposed at the Orsay museum.

VII. CONCLUSION

We have proposed a consistent model for artistic medium reproduction in 3D scenes and animations. Based on a consistent and coherent surface grain generation,

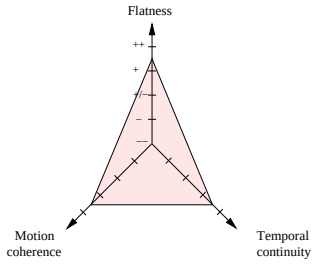

Figure 7. Diagram representation of the three goals for the additional material properties.

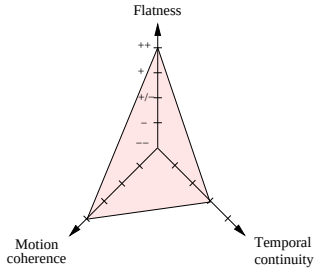

Figure 8. Diagram representation of the three goals for the additional material properties and fractal reconstruction surface.

produced with noise and/or textures, we create additional material properties used to produce stylization. We demonstrated that our parameter $\mathcal{T}$ permits to anchor a consistent noise / texture coordinates to the object and also to adapt the noise frequency / texture repetition according to the depth. As previously mentioned, our solution is a compromise and we favor the the temporal continuity and the coherence motion. In fact, a *resampling effect* could be perceived only through a z-axis (i.e. depth) displacement. But these additional material properties remain consistent according to the scene / objects animation while avoiding the *shower door* and popping effects. The fractal reconstruction surface can improve the flatness and the motion coherence to the detriment of the temporal continuity. In that case, the characteristics are extracted and used to reconstruct the whole image. Finally depending on the user choice, the consistent model for artistic medium can be realized through additional material properties and optionally fractal surface reconstruction.

In future work, we will investigate a fractal reconstruction surface designed for 2D animations (i.e. videos). Even if our fractal reconstruction surface gives accurate solutions, the reconstruction is realized according to each frame. As we provide consistent and coherent solution for grain surface generation, we would like to produce a coherent and consistent solution for the fractal surface generation and then improve its temporal continuity.

REFERENCES

[1] A. Petrov, “The old man and the sea,” Short Film, 1999.

[2] P. Litwinowicz, “Processing images and video for an impressionist effect,” in *SIGGRAPH ’97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1997, pp. 407–414.

[3] A. Hertzmann, “Non-photorealistic rendering and the science of art,” in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR ’10. ACM, 2010, pp. 147–157.

[4] P. Bénard, A. Lagae, P. Vangorp, S. Lefebvre, G. Drettakis, and J. Thollot, “A Dynamic Noise Primitive for Coherent Stylization,” *Computer Graphics Forum*, vol. 29, pp. 1497–1506, 2010.

[5] A. A. Gooch and B. Gooch, *Non-Photorealistic Rendering*. AK Peters, Ltd., 2001.

[6] T. Strothotte and S. Schlechtweg, *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*, Hardcover, Ed. Morgan Kaufmann, 2002.

[7] P. Bénard, A. Bousseau, and J. Thollot, “State-of-the-Art Report on Temporal Coherence for Stylized Animations,” *Computer Graphics Forum*, vol. 30, no. 8, pp. 2367–2386, Oct. 2011.

[8] B. J. Meier, “Painterly rendering for animation,” in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’96. ACM, 1996, pp. 477–484.

[9] M.-T. Chi and T.-Y. Lee, “Stylized and abstract painterly rendering system using a multiscale segmented sphere hierarchy,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 61–72, January 2006.

[10] M. Cunzi, J. Thollot, S. Paris, G. Debunne, J.-D. Gascuel, and F. Durand, “Dynamic canvas for immersive non-photorealistic walkthroughs,” june 2003.

[11] M. Kaplan and E. Cohen, “A generative model for dynamic canvas motion,” in *Computational Aesthetics*, 2005, pp. 49–56.

[12] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion, “Interactive watercolor rendering with temporal coherence and abstraction,” in *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ser. NPAR ’06. ACM, 2006, pp. 141–149.

[13] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin, “Video watercolorization using bidirectional texture advection,” in *ACM SIGGRAPH 2007 papers*, ser. SIGGRAPH ’07. ACM, 2007.

[14] D. Vanderhaeghe, P. Barla, J. Thollot, and F. X. Sillion, “Dynamic point distribution for stroke-based rendering,” in *Eurogaphics Symposium on Rendering*, Jan Kautz and Sumanta Pattanaik, Eds. Eurographics Association, 2007, pp. 139–146.

[15] M. Kass and D. Pesare, “Coherent noise for non-photorealistic rendering,” *ACM Trans. Graph.*, vol. 30, no. 4, p. 30, 2011.

[16] K. Perlin, “An image synthesizer,” in *SIGGRAPH ’85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. ACM Press, 1985, pp. 287–296.

[17] ——, “Improving noise,” in *SIGGRAPH ’02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM, 2002, pp. 681–682.

[18] P. Bénard, A. Bousseau, and J. Thollot, “Dynamic Solid Textures for Real-Time Coherent Stylization,” in *Symposium on Interactive 3D Graphics and Games (I3D)*. Boston, United States: ACM Press, 2009, pp. 121–127. [Online]. Available: http://hal.inria.fr/inria-00345835

[19] F. Belhadj and V. Boyer, “Fractal-based impressionist style rendering,” in *SITIS ’10: Proceedings of the 2010 IEEE International Conference on Signal Image Technology and Internet Based Systems*. IEEE Computer Society, 2010, pp. 60–65.

[20] F. Belhadj, “Terrain modeling: a constrained fractal model,” in *Proceedings of Afrigraph ’07*. ACM, 2007, pp. 197–204.

[21] J. Pouderoux, J.-C. Gonzato, I. Tobor, and P. Guitton, “Adaptive hierarchical rbf interpolation for creating smooth digital elevation models,” in *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, ser. GIS ’04. ACM, 2004, pp. 232–240.

[22] G. S. P. Miller, “The definition and rendering of terrain maps,” in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH ’86. ACM, 1986, pp. 39–48.

[23] P. J. Gärtner, *Art and Architecture, Musée d’Orsay*, Köenemann, Ed. Köenemann, 2000.

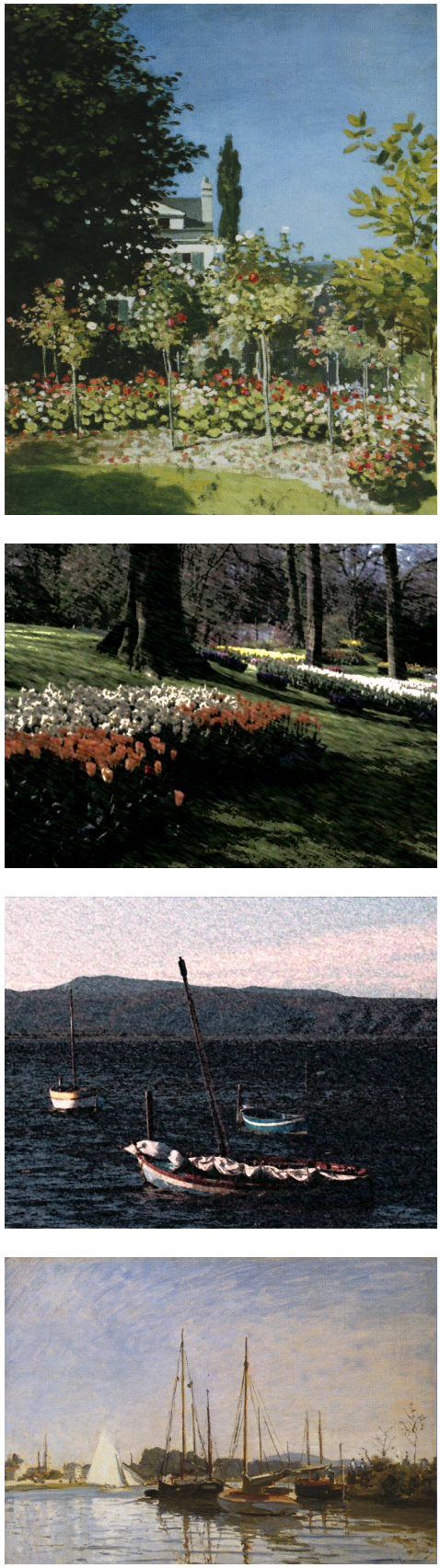

Figure 9. Four impressionist paintings. Some of them are original paintings realized by Monet, the others are produced by our model using the generated medium and the fractal reconstruction model.

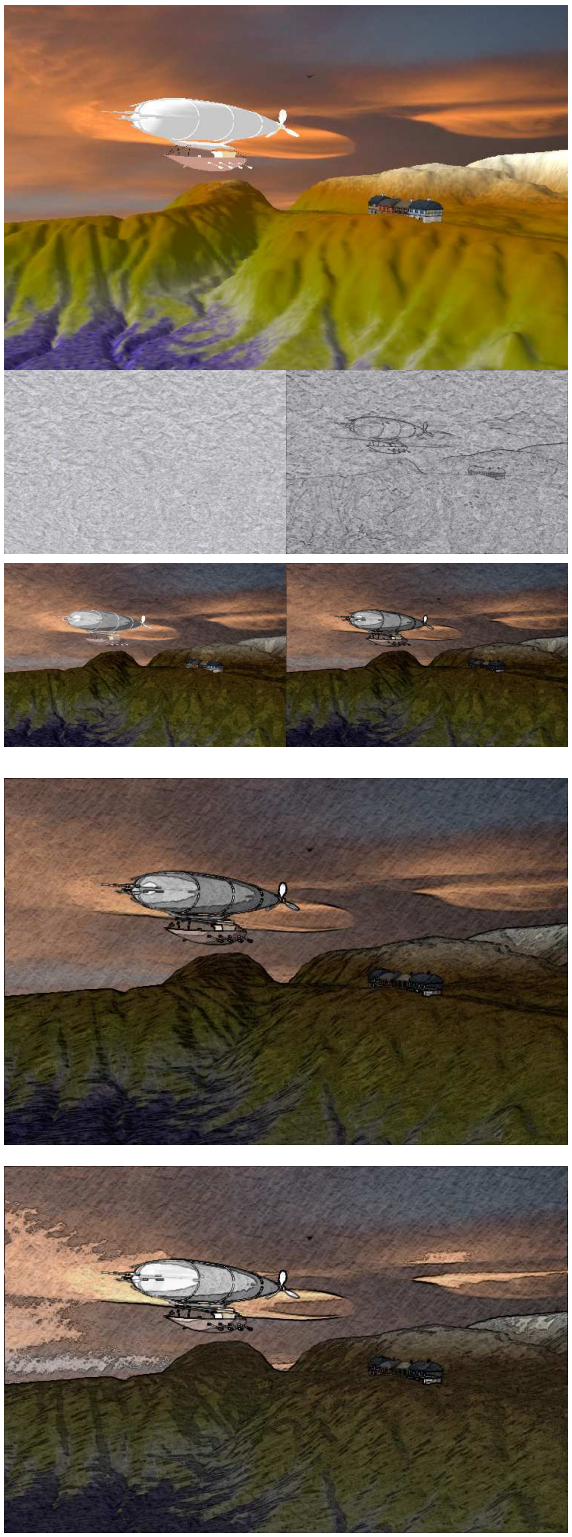

Figure 10. Various results obtained on 3D scene. From top to bottom and left to right: the original scene; noise generation without and with additional contours; application of the additional noise; noise variation (rotated and scaled); normal perturbation.

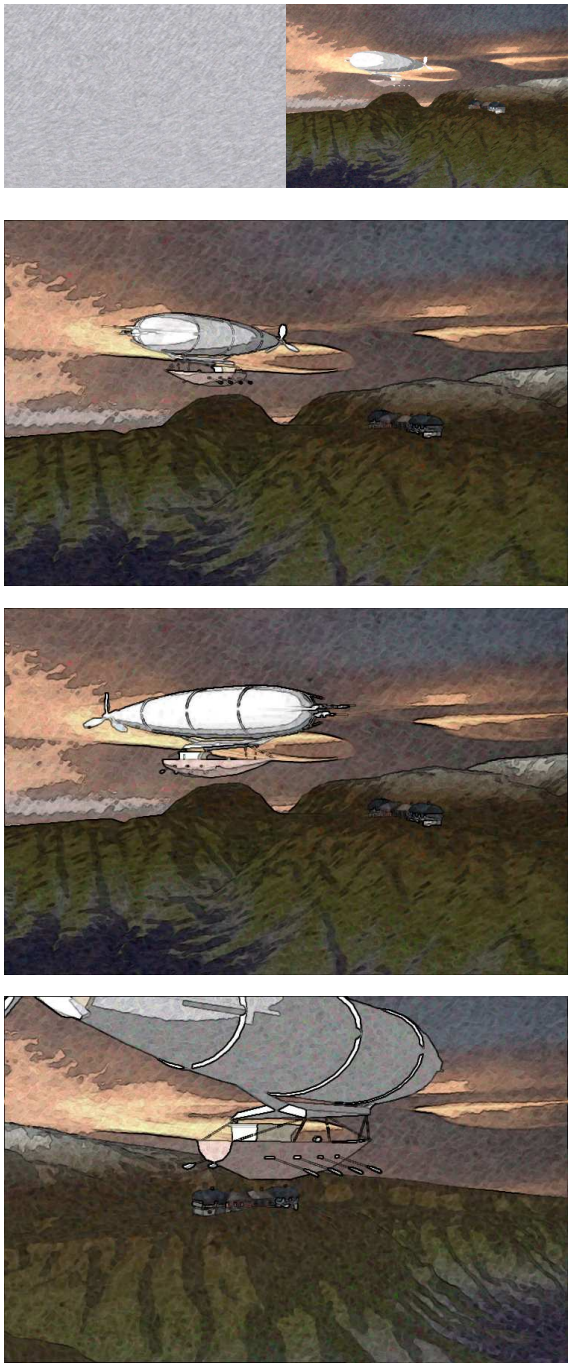

Figure 11. From top to bottom and left to right: reconstruction of noise after abstraction using MCMD; its application on the scene; three consecutive frames of an animation including fractal reconstruction and additional contours.