Towards Coherent Image Space Stylization of Animated 3D Shapes

Simon Breslav
MSc Report
Department of Computer Science
University of Toronto
May 2010

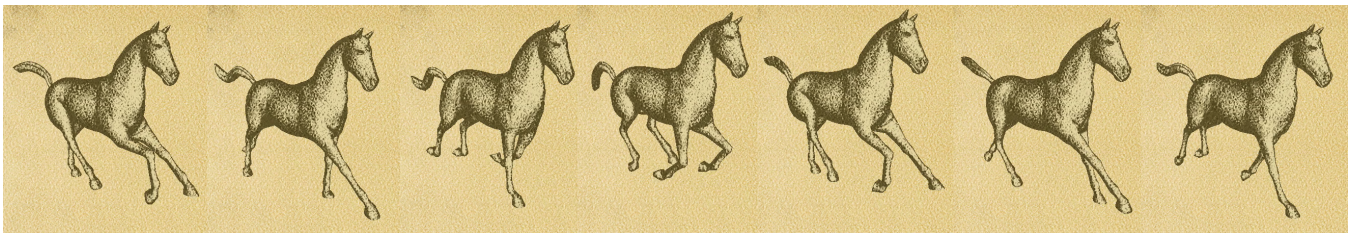


Figure 1: *A sequence of a horse running, demonstrating a highly deformable shape being rendered in a stippling style.*

Abstract

We describe a rendering technique for creating animations of 3D shapes in a range of non-photorealistic styles based on *2D patterns* that resemble hand-made drawings, prints, and paintings. Producing 3D animations of such styles presents a challenge: ensuring that the 2D patterns follow 3D motions while preserving their 2D appearance are conflicting goals. Our solution is to use texture synthesis in image space, optimizing a simple objective function that trade-off error of both 2D structure of the example pattern, and matching 3D motion. This method provides a good solution for wide variety of styles, ensures the preservation of 2D characteristics of the pattern, provides a good temporal coherence, and matches desired orientations for anisotropic patterns.

1 Introduction

Non-photorealistic renderings of 3D scenes depict the computer-generated geometry as an artistic creation, that resembles hand-made drawings, prints, or paintings. As with artistic imagery, different rendering styles help convey distinct feelings to the viewer, and the perception of tone, shape and material can be enhanced by an expressive, non-photorealistic rendering (NPR). However, hand-made artwork is typically confined to the 2D static space of a canvas. Extending an artistic style to 3D dynamic scenes poses the problem of illustrating the 3D movement of the scene, while maintaining the 2D characteristics of the style. In the current approach, we aim to satisfying these constraints for a large variety of styles. This imposes a set of conflicting goals for our method:

1. The *2D patterns* (strokes, pigments, etc.) that convey the artistic style should have a constant size and density in the image. This ensures that the style preserves its 2D appearance, and does not appear directly drawn on the 3D objects.

2. The *2D patterns* should follow the motion of the 3D objects they depict.

As well as these desired properties:

- Often these 2D patterns, (strokes in particular) are organized to follow the features of the 3D shapes that make up the scene. This helps convey a better sense of 3D shape to the viewer [Girshick et al. 2000]. So another optional constraint can be to match a desired orientation defined on the surface of the shape.

- Since many of the target styles are a particularly designed to convey tone (i.e.hatching, halftoning), another constraint is that resulting stylized image should match the perceived tone of the underlying 3D shape.

- To support a wide range of styles, our method should not rely on characteristics typical to a particular style.

To comply with these constraints, our solution is to use example-based *Anisotropic Texture Synthesis* [Ying et al. 2001; Lefebvre and Hoppe 2006] in image space to generate the stylized rendering of the shapes. We formulate the method as a minimization problem with two main terms, the *Texture* term and the *Coherence* term. The *Texture* term guarantees that the structural content of the style is preserved when each still frame is computed, and the *Coherence* term ensures that a given frame is consistent with previous or next frames.

Our main contribution is, therefore, to demonstrate that image space example-based texture synthesis approach for rendering non-photorealistic patterns is a flexible way to satisfy all our objectives for a large variety of styles with a single unified policy, while supporting shape deformations and any camera motions (translation, rotation, and zoom).

2 Related Work

A number of previous techniques focused on representing 3D animated scenes with non-photorealistic styles.

To solve the contradiction between the 2D characteristics of a style and the necessity of following a 3D motion, *stroke-based* rendering methods [Hertzmann 2003] represent the style as a collection of stroke marks. The stroke appearance is rendered in 2D, but each stroke is attached to points on the 3D surface, and thus follow the scene motion. The decomposition of a style into strokes requires capturing the stroke particularities, and has given rise to methods that focus on one individual style: painterly [Meier 1996; Snavely et al. 2006], watercolor [Bousseau et al. 2006], and stippling [Pastor et al. 2003; Vanderhaeghe et al. 2007]. A general method that captures many artistic styles would be difficult to create. Also, stroke-based rendering requires the blend-in/blend-out of appearing

and disappearing elements. Texture synthesis on the other hand, allows for greater style diversity and avoids blending by texture 'morphing' of one element into another.

An alternative to stroke-based methods is texture mapping, in which the style is represented by a texture, and treated as a continuous 'data'. Two possibilities exist when using texture: either attach the stylization texture directly onto the 3D objects (object-space texture mapping), or keep the texture in 2D and rely on 2D transformations to suggest the 3D motion (screen-space texture mapping).

When considering the two conflicting constraints, matching 3D motion while preserving 2D characteristic of the pattern, *object-space* methods favor matching 3D motion over the 2D appearance of the style. Approaches like [Klein et al. 2000] and [Praun et al. 2001] perfectly follow the 3D animation, but the style texture is severely distorted by the perspective projection. Bénard *et al.* [2009] achieves a uniform image space scale of the object-space mapped style by combining different frequencies of the texture. Because the method blends together several texture frequencies, it is aimed at self-similar textures, and not suited for more structured textures. Moreover, the method does not allow the texture elements to be oriented along the surface; therefore, important shape cues, such as the surface curvature, cannot be represented.

The opposite approach, that guarantees the preservation of 2D characteristics, at the expense of temporal artifacts, is taken by *screen-space texture mapping* methods. For the specific case of line-art illustrations, [Kim et al. 2008] use screen-space principal curvature directions to illustrate 3D shape in dynamic and specular scenes, but their method is not equipped with explicit handling of 3D motion. [Cunzi et al. 2003] approximate the camera 3D motion with 2D transformations of the texture; camera translations in depth are represented with infinite zoom method that maintains constant the size of the texture. Because the motion is attached to the camera, and not the scene, this method works best in the limited case of a static scene in which all objects lie at roughly the same distance from the camera.

[Coconu et al. 2006] extract a similarity transform by tracking a single 3D point and orientation vector on the surface of "high level primitives" fit to the geometry. Similarly, [Breslav et al. 2007] uses a collection of 3D sample points and weighted least-squares optimization to compute a similarity transform to approximate the 3D motion. With both [Coconu et al. 2006] and [Breslav et al. 2007], the movement of large areas of the surface is approximated with same 2D transformation, leading to sliding effects for extreme motion. Also, both methods cannot follow a given orientation field as our method can. To minimize this sliding [Breslav et al. 2007] segments the shape into patches and find a transform for the individual patch, then blends the pattern at batch boundaries. This does work well for some patterns, but blending may not work for all target patterns, also, distracting asynchronous level of detail transitions are possible.

Bousseau *et al.* [2006] and Kaplan and Cohen [2005] also describe methods for achieving a temporally coherent dynamic canvas. These methods are automatic and account for camera motion as well as objects that move independently. Both methods track seed points on 3D surfaces and use them to generate (each frame) a canvas texture by joining small pieces of texture that follow the seed points. These methods work well with small-scale, unstructured patterns like canvas and paper textures, but cannot preserve regular or large-scale patterns like hatching and halftone screens. Eissele *et al.* [2004] propose a similar strategy for 2D halftone patterns applied to 3D scenes. Although this method shares some similarity to ours, the method exhibits severe temporal artifacts, and does not support oriented patterns or different styles.

Our approach is inspired by work in the area of texture synthesis. Great advancements have been achieved in the area in the recent years, resumed in the survey of Wei *et al.* [2009]. We were particularly influenced by the works of [Kwatra et al. 2007] and [Narain et al. 2007] on texturing fluids. Whereas their methods synthesize textures in object space, we address the specific NPR problem of maintaining a 2D aspect for the synthesized texture during the entire 3D animation. Work on Anisotropic Texture Synthesis also resembles our approach, but previous methods either dealt with static images [Wang et al. 2004; Taponecco et al. 2006], or synthesised the texture directly on the surface of 3D shapes [Ying et al. 2001; Lefebvre and Hoppe 2006; **?**; **?**]. Our approach, on the other hand, considers animated 2D shapes, and targets a different class of patterns the most texture synthesis work. While there has been work that demonstrated using texture synthesis to generate non-photorealistic images (i.e.[Hertzmann et al. 2001; Wang et al. 2004]), to our knowledge no work has yet done so for non-photorealistic animations.

3 Overview

Our method is broken into three stages. In the first stage, Fields and Data Extraction (see Section 4), we project all the necessary information from the 3D shape into images space. The shape information we process consists of: the tone (i.e. a diffuse rendering of a 3D shape), the mask (that indicates the area occupied by the shape), the edge map (that captures the view-dependent shape discontinuities and creases), the occlusion/dis-occlusion map, the orientations, and the backwards and forward optical flow fields. Figure 2 illustrates the entire set of extracted data. During the first stage, we also pre-process the *Example Pattern* P that will be used for texturing the shape. From P, we extract three features: the Distance To Edges, Image Gradients, and the Orientations (Figure 4) and we project them into the Appearance-Space $\tilde{P}$ proposed by [Lefebvre and Hoppe 2006].

In the second stage (see Section 5) all the data, fields, and the *Example Pattern* from the first stage are used by our multi-scale *Texture Synthesis* algorithm that produces either a halftone screen or a layer of paint. Our method is formulated as an optimization problem, where we minimize a texture energy function that measures the extent to which synthesized image deviates from both *Example Pattern*, and from the *advected* previous frame.

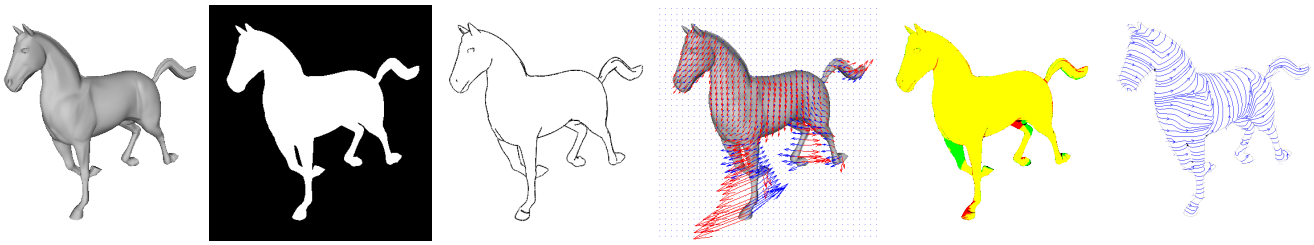
As a final stage (see Section 6), we apply various post-processing operations, such as halftone thresholding, paper medium simulation, tone correction, and non-uniform shape boundary erosion.

4 Fields and Data Extraction

Given a 3D shape, we output for each animation frame, the 2D image of tone, mask, feature lines, occlusion/dis-occlusion map, orientations, and backwards and forward optical flow fields (See Figure 2 for a frame of this data). Most of this data is rendered at multiple scales (i.e 720x480, 360x240, 180x120) to assist our multi-scale Texture Synthesis algorithm. Re-rending our 3D scene on multiple scales give us higher precision then down-sampling the information, especially in cases such as optical flow flow fields.

4.1 Feature Lines

For our feature lines (See Figure 2 (c)) we use silhouettes and crease edges. These feature lines are used first during our texture synthesis process to stop the texture from crossing occlusion boundaries, and then to highlights discontinuities in our final result.



(a) Tone (b) Mask (c) Feature Lines (d) Optical Flow (e) (Dis-)Occlusions (f) Orientations

Figure 2: Data Extracted from 3D. *(d)* Green*: Dis-occlusions (were occluded in the last frame),* Red*: Occlusions(will be occluded in the next frame),* Yellow*: Visible in previous and next frame (e) Orientations based on derivative of UV coordinates (f) Red: Backwards Optical Flow(where was the vertex in the last frame), Blue: Forward Optical Flow(where will the vertex be in the next frame).*

Silhouetts are determined using algorithm described in Markosian *et al.* [1997]. To find creases, every edge of the 3D mesh is visited and is labeled a creases if dot product between normals of two adjacent faces exceeds a threshold (i.e. 0.5). All feature lines are drawn using GL line strips.

4.2 Optical Flow

The optical flow is computed at each vertex on the mesh. Vertex is project into image space in each frame, and pixel difference of the projected location between adjacent frames is used. This is done both for forward and backwards frames. Barycentric interpolation is then used to get per-pixel flow. See Figure 2 (d) for a quiver plot of the optical flow field at a given frame.

4.3 (Dis-)Occlusion Map

Occlusion and Dis-occlusion Maps are used as a optical flow reliability map. If a vertex is visible in a previous frame, it's backwards optical flow is reliable, if it will be visible in the next frame, then it's forward optical flow is reliable. Similarly to optical flow, barycentric interpolation is used for getting per-pixel values from per-vertex visibility information. In Figure 2 (e) red denotes pixels that will be occluded in the next frame, green marks dis-occluded pixels (pixels that were occluded in the last frame), and yellow shows pixels that were visible in last frame, and will be visible in the next. The actual visibility is done using "ID image" visibility method described in Kalnins *et al.* [2002].

4.4 Orientations

Our method supports arbitrary orientations, however to achieve good looking results, spatially and temporary smooth fields without too many singularities or orientation flips are desired. To achieve this we experiment with following fields:

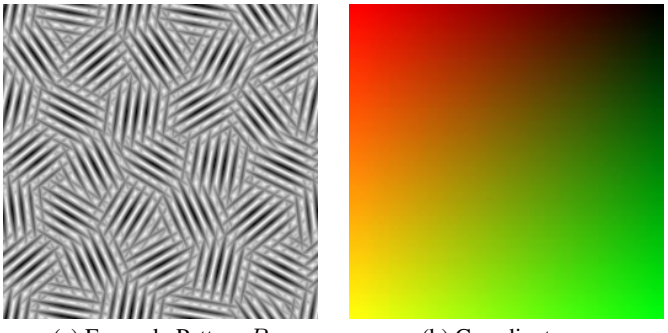
Constant Uniform 2D orientation may be useful for abstracting the shape (i.e. Artist's quick sketching at 45^o). Using these uniform orientations does pose a challenge, if the camera rotates or the shape is deformed, there will be a disconnect between local orientation change of the region and the constant orientations. This disconnect introduces artifacts to the temporal and spatial coherence in the final textured results. Nevertheless, in many cases our method handles these difficulties with satisfactory results.

UV Based Given a 3D shape with UV parametrization, we can use the derivative of UV coordinates at each vertex as our orientation. To find this orientation, we first calculate UV coordinate derivative at each face, with respect to $\vec{x}$, $\vec{y}$, $\vec{z}$. Then we take the average of all the neighbouring face directions projected into local coordinates of the vertex defined by the $\vec{b_1} = \vec{n} \times (1,1,0)$, $\vec{b_2} = \vec{n} \times \vec{b_1}$, where $\vec{n}$ is the vertex normal. The resulting 3D vector is then projected into image plane to get our final 2D orientation. Using orientations extracted from UV coordinates allows us to have necessary control to define orientations where automatic methods fail. For example, principal curvature directions are not defined on planer regions. Also, it's worth noting that requirements for the quality of the parametrization for orientation extraction are a bit different then for general purpose UV parametrization. For purposes of traditional texture mapping stretching of UV coordinates is undesired, in our case, we are only concerned with orientation quality, thus excessive stretching is not a factor.

View-Dependant As described in [Markosian et al. 1997] we can used directions defined by the cross product of local surface normal and the ray from the camera. While these orientations create singularities when the surface normal and the camera ray are parallel, the resulting orientations are quite smooth. Since these orientations are view dependant they are not temporally coherent with respect to shape deformation and will slide on the surface when the camera rotates. Nevertheless, as with constant orientations in certain styles our method handles these difficulties with satisfactory results.

In the future work we would like to take advantage of large body of work that use automatic orientation generation on the surfaces (i.e. [Hertzmann and Zorin 2000], [Turk 2001], [Wei and Levoy 2001], [Praun et al. 2001], [Fisher et al. 2007], [Xu et al. 2009]).

4.5 Example Pattern Pre-Processing



(a) Example Pattern P (b) Coordinates u

Figure 3: *Input Example Pattern*

As noted by Durand *et al.* [2001], it is desirable for the halftone screen to have flat histogram. Thus, as a first step to preparing our example patterns, we apply Contrast-limited adaptive histogram equalization (CLAHE)[Zuiderveld 1994] to our patterns. See

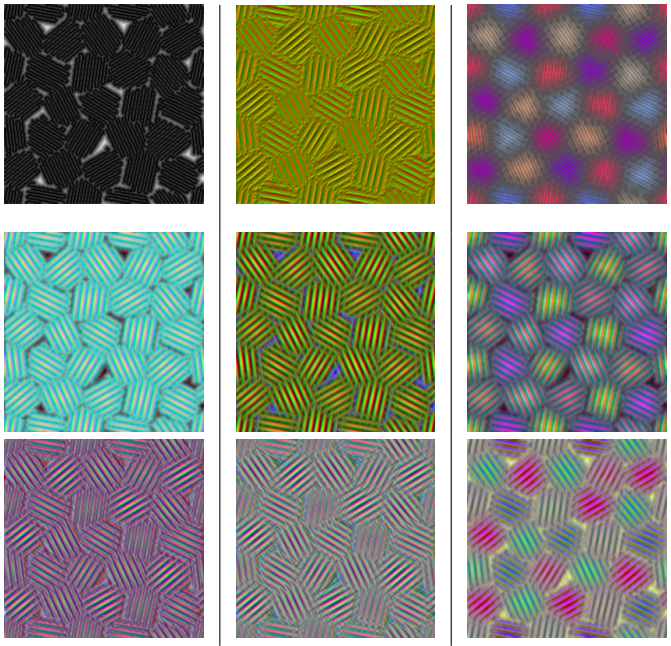


Figure 4: *Top Row: 6 channels of $\acute{P}$ (a) Distance To Edges (1 channel) (b) Image Gradient (2 channel) (c) Orientations (3 channel). Middle Row: First 3 channels of $\tilde{P}$ when the corresponding feature (top row) is added. Bottom Row: Channels 4-6 of $\tilde{P}$.*

Figure 5 for result of applying this equilibration, as well differences in the results of the synthesis, and the thresholding.

Since most of our example patterns are grayscale, in many cases we find it useful to extract certain features to improve results of the synthesis. All the extracted features are combined with P to be used as a new example pattern, $\acute{P}$. Here is a list of features that we optionally add:

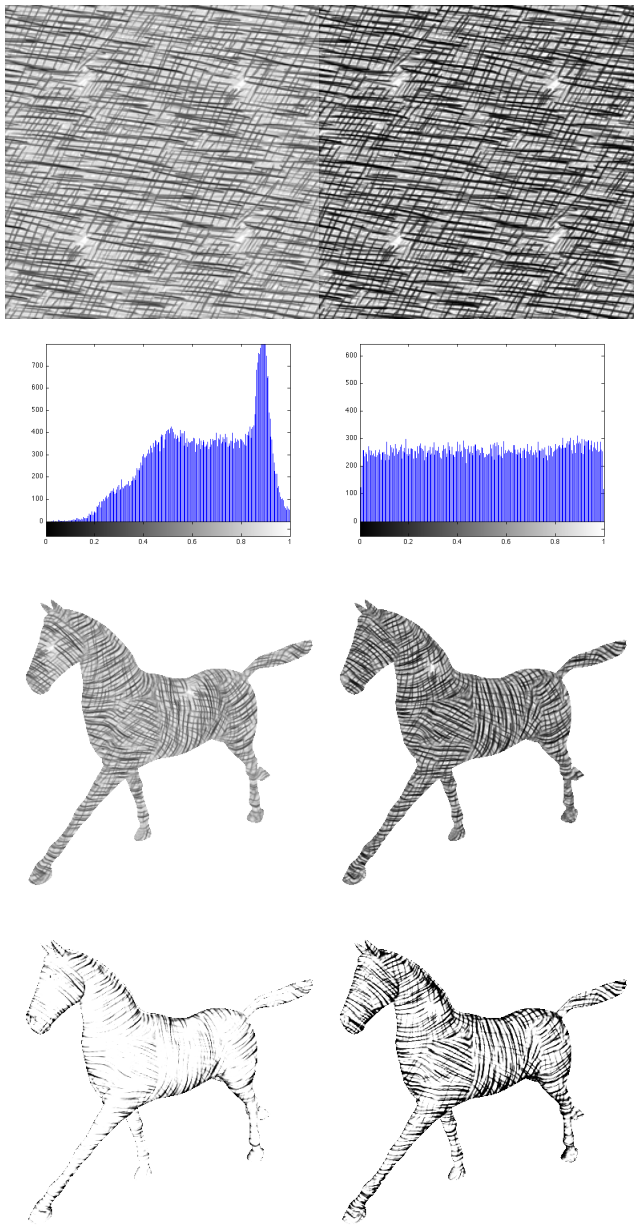
Distance To Edges We extract edges using Canny Edge detection [Canny 1987], then apply Euclidean distance transform [Breu et al. 1995](see top row of Figure 4 (a).) This feature is consistent with previous work such as Lefebvre *et al.* [2006], although may not be that descriptive on the dense edges as seen in example texture used in Figure 4.

Image Gradient We extract image gradient, $\frac{\partial P}{\partial x}$, $\frac{\partial P}{\partial y}$, using Sobel operator.

Orientations To distinguish areas of different orientation, we use 2D Structure Tensor, also referred to as the "second-moment matrix", which is commonly used for corner detection [Harris and Stephens 1988]. For our feature we use the values of this matrix, $(\frac{\partial P}{\partial x})^2$, $\frac{\partial P}{\partial x}\frac{\partial P}{\partial y}$, $(\frac{\partial P}{\partial y})^2$ as additional three channels (see top row of Figure 4 (c)).

(Note, if P is not grayscale, we convert it to grayscale before extracting these features.)

To increase the density of the information and to allow for the use of smaller neighbourhoods we project out pattern combined with all the features, $\acute{P}$, into Appearance-Space [Lefebvre and Hoppe 2006], $\tilde{P}$. To do so, let the Gaussian-weighted 5x5 neighbourhoods of $\acute{P}$ define a maximum of 175D (for grayscale patterns, and 225D for color patterns) set of points. We then reduce the dimensionality of these points using PCA to obtain a 5D to 8D points, where each point corresponds to a pixel in P, and will be used as our new



(a) Original Pattern (b) Equilibrated Pattern

Figure 5: *Pattern Histogram Equilibration. Row 1: Example Pattern. Row 2: Histogram of intensities. Row 3: Resulted Halftone Screen. Row 4: Thresholded Result.*

example pattern for synthesis, $\tilde{P}$.

As in Lefebvre *et al.* [Lefebvre and Hoppe 2005] we associate Gaussian Stack $\tilde{P}_0, \tilde{P}_1, ..., \tilde{P}_L$ with our Example Pattern to work with our multi-scale texture synthesis.

5 Approach

After all the data has been extracted from the 3D, we proceed to synthesize our pattern in images space. Our algorithm is inspired by existing *Texture Synthesis* methods [Ying et al. 2001; Ashikhmin 2001; Lefebvre and Hoppe 2006; Narain et al. 2007]. The synthesis

process involves creating an an image S, or more precisely an image pyramid $S_L, S_{L-1}, ..., S_0$ (Figure 9 (e-h)) in coarse-to-fine order (typically $L = 2$ in all our examples, resulting 3 pyramid levels). Each pixel in S stores the coordinates of the Example Pattern (Figure 3). Thus, a color at a given pixel i, in the result image is given by $P[S[i]]$, although please see Section 5.3 for details about a better mapping function that hides patch boundary seams.

5.1 Energy Function

The output texture is generated by minimizing an energy function $\mathbf{E}_{texture}(s; p)$ which measures the similarity of each $n \times n$ patch s in the result image with a similar $n \times n$ patch p in the input Example Pattern. For temporal coherence, we minimize an energy function $\mathbf{E}_{coherence}(s; s')$, which measures similarity of each $n \times n$ patch s in the current frame, with the corresponding $n \times n$ patch s' in the previous frame. The net energy function for an image patch s is:

$$\mathbf{E}(s) = \alpha \mathbf{E}_{texture}(s; p) + \beta \mathbf{E}_{coherence}(s; s') \quad (1)$$

α and β are user specified weights to manipulate dominance of each term (usually just 1.0 and 1.0). Specifically, the two terms take the following form:

$$\mathbf{E}_{texture}(s; p) = \|\tilde{I}(s) - \tilde{I}(p)\|^2 \quad (2)$$

$$\mathbf{E}_{coherence}(s; s') = \|\tilde{I}(s) - \tilde{I}(s')\|^2 \quad (3)$$

Where $\tilde{I}$ refers to the feature pixels associated with a patch. The total energy over all patches $\sum_S \mathbf{E}(s)$, can be minimized by an iterative approach; we refer the reader to previous work for details [Kwatra et al. 2005; Kwatra et al. 2007; Han et al. 2006; Narain et al. 2007]. We do not perform such global optimization, and just minimize error locally. For Anisotropic Synthesis, our patches have to be deformed to align with the orientation field (see example field in Figure 2 (f)), we refer the reader to previous work for details [Ying et al. 2001; Lefebvre and Hoppe 2006; Eisenacher and Lefebvre 2008]. In case of isotropic synthesis, patches don't have to be deformed, and we perform regular patch sampling.

5.2 Optimization Algorithm

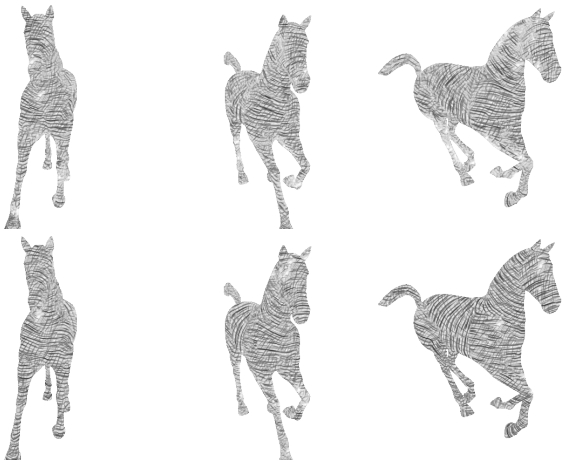
We aim at generating a resulting image by following a sequence of operations: first, we *initialize* the coarse scale of this result pyramid by covering the image with overlapping patches (Section 5.2.2) followed by a step of *correction* (Section 5.2.4). Then, for each remaining pyramid level, we perform *upsampling* (Section 5.2.3) and *correction* (Section 5.2.4).

This is performed for each frame, where $\mathbf{E}_{texture}(s; p)$ is used to evalute texture error for each patch s, and $\mathbf{E}_{coherence}(s; s')$ for patches s that were not occluded in the last frame, *Dis-Occlusions Map* (see Figure 2 (e)) is used to decide if coherence term should be used or not.

5.2.1 Bi-Directional Synthesis Loop

When the object rotates around an axis that is parallel to the film plane (e.g. Figure 7), in each frame there will be a sliver of pixels that were occluded in the last frame. Because this sliver is near an edge, only certain portion of the patch will be valid for comparison, thus having a less descriptive match. In the subsequent frames, as that area becomes less foreshortened and grows, these local minimums might propagate and create undesired artefacts.

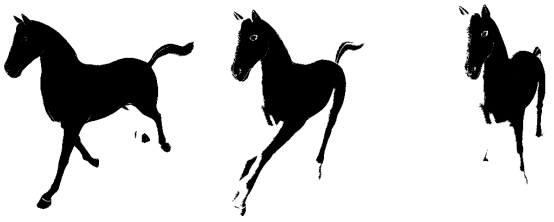
To combat this problem, we introduce a *Bi-Directional Loop* that take advantage of the fact that we extract both forward and backwards optical flow from 3D. See Figure 7 for effects of using *Bi-Directional Loop*.



(a) Frame 20 (b) Frame 30 (c) Frame 40

Figure 7: *Top Row: Regular Synthesis Loop. Bottom Row: Bi-Directional Loop.*

First we extract an addition piece of information from 3D, *Delayed Dis-Occlusions Map*, see Figure 8 for a couple of example frames. To create this *Delayed Dis-Occlusions Map*, instead of updating *Dis-Occlusions* each frame, pixels that were visible last frame, are only updated in every x frames (we use around $x = 20$). In every frame other then the *update frame*, we invalidated pixels in subsequent frames if they become dissociated.



(a) Frame 2 (c) Frame 10 (d) Frame 15

Figure 8: *Delayed Dis-Occlusions. Black indicates the point was visible in in current frame and to last* update *frame.*

To Use *Delayed Dis-Occlusions* instead of regular Dis-Occlusions, we modify the synthesis loop as follows:

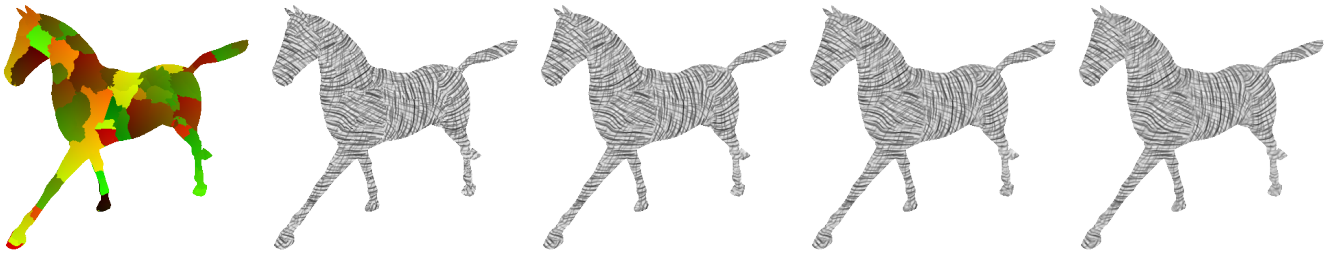
Step 0 Fill Frame 1 only using $\mathbf{E}_{texture}(s; p)$ (same as before).

Step 1 Synthesize going forward in time, only in areas where Delayed Dis-Occlusions is valid (Black in Figure 8). For all these pixels we can use $\mathbf{E}_{texture}(s; p)$ and $\mathbf{E}_{coherence}(s; s')$.

Step 2 In Frame x fill the whole shape, some areas may only use $\mathbf{E}_{texture}(s; p)$.

Step 3 Go backwards x frames in time; using forward optical flow field instead of backwards, and fill in unfilled areas. Again, both $\mathbf{E}_{texture}(s; p)$ and $\mathbf{E}_{coherence}(s; s')$ will be used, except now the coherence term point forward in time rather then backwards.

Step 4 Skip x frames forward and Go to Step 1



(a) Coords (a) No Blending (b) 3x3 (c) 5x5 (c) 7x7

Figure 6: *Coordinate Blending.*

5.2.2 Initialization

To initialize the coarse pyramid scale, we extend Efros and Freeman [2001] with each patch being deformed using a dense orientation field, the same as was done in previous anisotropic texture synthesis methods [Ying et al. 2001; Lefebvre and Hoppe 2006]. Each patch consists of $n \times n$ pixels (for our examples we use n of 10-14 pixels). We fill the area one patch at a time using either scanline order, or custom fill order based on morphological skeleton of the fill area. As we traverse fill order locations looking for the next patch center location, we skip already filled pixels as new patch center, this policy gives an average overlap of $n/2$ pixels between the patches.

We do not preform minimum error boundary cut as was done by Efros and Freeman[2001], and simple overwrite $\tilde{S}_L$ with values of the new patch. This is done for simplicity of the algorithm, and because the *correction* stage takes care of correcting the boundaries.

To find the best patch for each point, we use the objective function described in Section 5.1, where $p \in C$, with C being the candidate patch set, in case of initialization, we use exhaustive list of patches in $\tilde{P}_L$ (coarse level of example pattern pyramid) with 2^L spacing.

5.2.3 Upsampling

To create a finer image from the next-coarser level, we upsample the coordinates of the parent pixels. Similarly to Lefebvre and Hoppe [2005] we assign to each of the four children the scaled parent coordinates plus a child-dependent offset. To account for anisotropic synthesis, the offsets have to be deformed using the deformation field, this is consistent to how patches were deformed during evaluation of our objective function. The process can be summarized by the following formula:

$$S_l[2i + \Delta] = S_{l+1}[i] + \phi(i; \Delta)h_l \tag{4}$$

Where Δ is a set of offsets, $\left\{\binom{0}{0}\binom{0}{1}\binom{1}{0}\binom{1}{1}\right\}$, h_l is the spacing on a given scale of the pyramid, so is equal to 2^l, and $\phi(i; \Delta)$ is a deformed offset $R_l[2i+\Delta]\Delta$ for anisotropic synthesis, where R is the deformation field, sampled at pixel i, this is the same field used with deforming Example Pattern neighbourhoods. For isotropic texture synthesis $\phi(i; \Delta)$ is just Δ.

5.2.4 Correction

The correction step takes the existing coordinates and tries to find better neighbourhoods that minimize our overall objective function, equation (1).

To accelerate neighbourhood matching we use *coherence synthesis* [Ashikhmin 2001], only considering those locations in the Example Pattern given by a 3×3 immediate neighbours of a image pixel location i in S. Where i is also a center pixel of a given patch s we are minimizing error for.

If the backwards optical flow field if valid (i.e. Yellow and Red in Figure 2 (e)), we also include candidates around pixel i', which represents location of pixel i was in the last frame's S. Since i' coordinate might be a real number, we use bilinear interpolation on all the fields to come up with candidate coordinates. Since S has coordinate discontinuities, interpolating coordinates does not always makes sense, however, since we take a neighbourhood of candidate coordinates, we are bound to have good patch matches.

After the whole animation sequence has been animated, we can repeat this correction process, and in the areas with valid forward optical flow (i.e. Yellow and Green in Figure 2 (e)) also use i'' where i'' is where i will be in the next frame's S.

5.3 Blending Coordinates

As mentioned earlier, we synthesize image S in which each pixel i in S stores the coordinate of the Example Pattern pixel. The resulting color at each pixel is given $P[S[i]]$. However, S may have undesirable boundaries, thus we find it useful to blend patches along the boundaries using following formula:

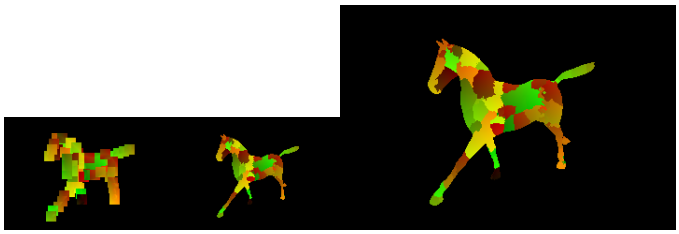
$$I[i] = \frac{1}{N} \sum_{\Delta \in \{-1..1\}^2} P[S[i + \Delta] - \varphi(i; \Delta)h_l] \tag{5}$$

Where Δ is a 3×3 neighbourhood offsets, $\varphi(s; \Delta) = R_l[i + \Delta]\Delta$ are the deformed offsets to alight with the our orientation field, where R is the deformation field, $h_l = 2^l$ is spacing at a given pyramid level, and N is the number of points in the neighbourhood patch. Note, that if i is not on a coordinate boundaries, the resulting color will not be changed due to the appropriate offsets.

6 Post-processing

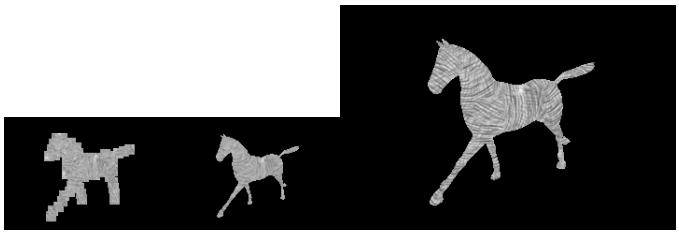
Texture synthesis produces either a halftone screen, or a layer of paint. To achieve the final look, there are number of post processing operations that we perform. In case of halftone styles, we uses a halftone thresholding method [Ostromoukhov and Hersch 1999; Durand et al. 2001; Freudenberg et al. 2002]. To produce more pancil-like rendering, we use media simulation method [Kalnins et al. 2002] to give appearance of paint being applied on paper. The texture of paper is synthesized the same way the halftone/paint texture, to maintain temporal coherence. Decoupling texture from

tonal adjustments allows for a greater control of the lighting, without the need to recompute the texture layer, where density needs to be adjusted.

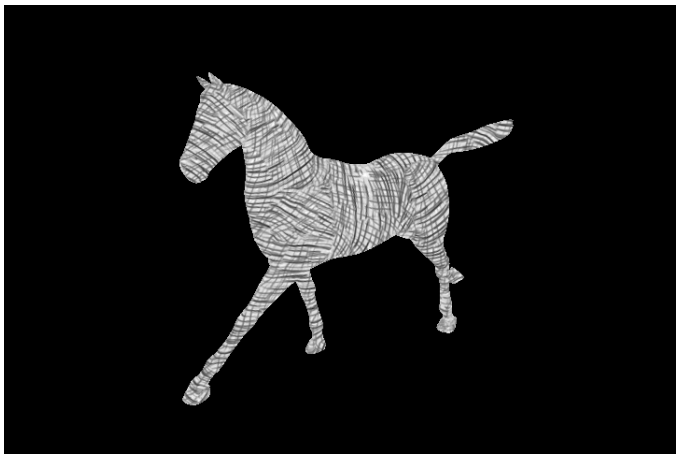


(a)Initial S_2 (b) Corrected S_2 (c) S_1

(d) S_0



(e) Initial I_2 (f) Corrected I_2 (g) I_1



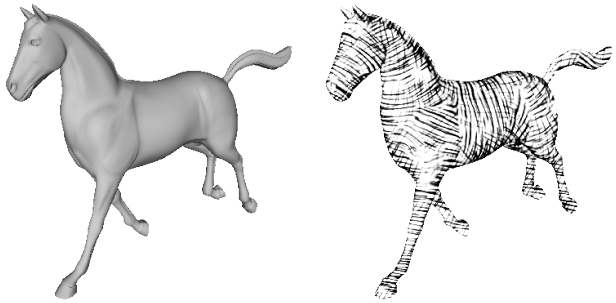
(h) I_0

Figure 9: *(a-d) S Image Pyramid (e-h) I Image Pyramid*

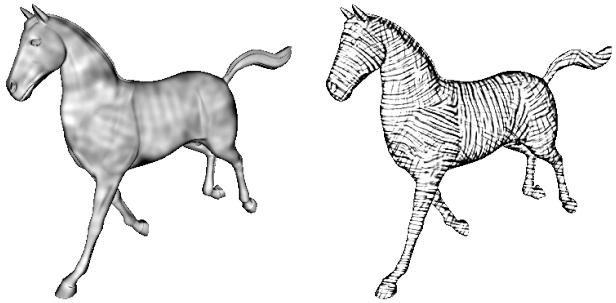
6.0.1 Tone-Correction

Despite the histogram normalization described in Section 4.5, there is still a possibility of uneven tone in the resulting thresholded result. This can be due to blurring in the halftone screen, uneven intensity distribution in the example pattern, or other artefacts of texture synthesis. To address these issues we adapt a similar solution to previous work [Salisbury et al. 1997; Ostromoukhov and Hersch 1999; Durand et al. 2001].

After the initial thresholding operation, the result, Figure 10 (b), is blurred (to simulate spatial integration of the visual system [Ostromoukhov and Hersch 1999]), and subtracted from original tone, Figure 10 (a), to get our a new tone to be used, Figure 10 (c). This process can be repeated, yet we find it, just one time is enough to get a pleasing result. The result of using the new adjusted tone can be seen in Figure 10 (d). Notice a more even stroke density, and a resulting tone matching the original tone better.



(a) Initial Tone (b) Initial Result



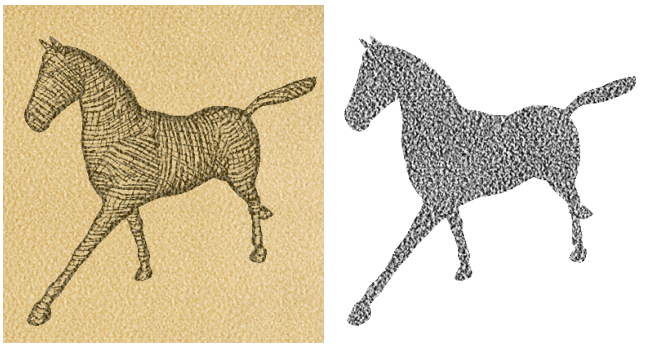
(c) Adjusted Tone (d) Adjusted Result

Figure 10: Tone Correction. *Threshold result (b) is blurred and subtracted from original tone (a), to get our a new tone (c). The result of using the new adjusted tone (c) can be seen in (d).*

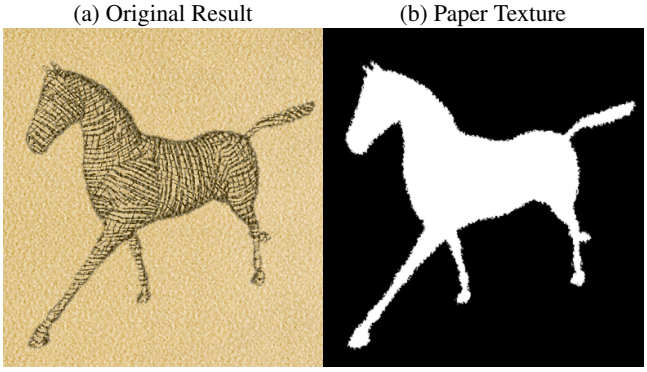
6.0.2 Shape Boundaries

Since the source of your renderings are 3d shapes, the shape boundaries are much smoother then hand-drawn images. While for some style it's quite acceptable, and even desired, for effects that involve paper effect, it may look too mechanical. Ideally we would try to apply a method equivalent to stroke-based coherent stylized silhouettes method [Kalnins et al. 2003], but doing so within our framework is outside the scope of this work.

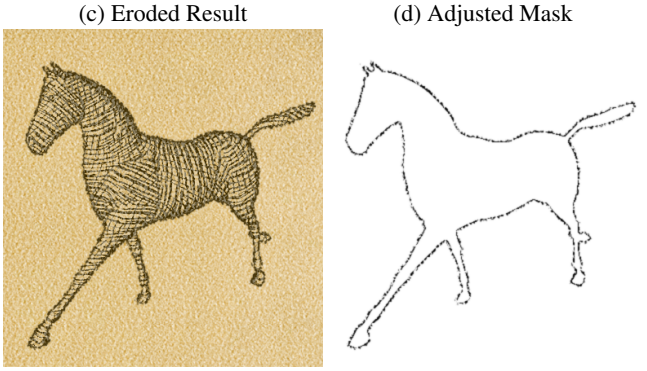
Instead we propose a simple limited solution to this problem. We erode the shape mask (Figure 11 (d)) in a non-uniform way to make boundaries less smooth. The noise that is used to erode the boundaries comes from paper texture layer. Because this paper



(a) Original Result (b) Paper Texture



(c) Eroded Result (d) Adjusted Mask



(e) Result with Silhouettes (f) Silhouette Line

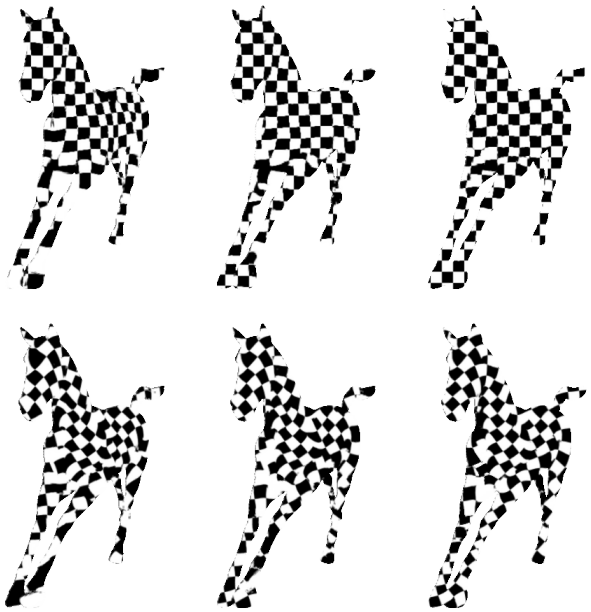
Figure 11: *Boundary Erosion.*

texture layer is also synthesised the same way as our halftone screen layer, it is temporally coherent.

Eroding the edges, may result in the loss of dark boundary and objecting mixing with the background too much (see Figure 11 (c)), to fix that, we simply add a a bit of dark silhouette paint at the boundaries of the shape (Figure 11 (f)), resulting in a more defined shape silhouette.

7 Results

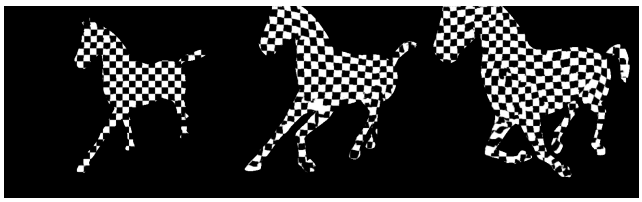
Formulating the problem as an optimization problem allows for some interesting interactions. If we modify weights of the term in the objective function, we can get different behaviour. See Figure 12 for effects of changing these weights. If texture term weight is decreased, then we get a more coherent result, yet the texture gets more distorted, on the other hand if we make the coherence term weight smaller, the texture is less distorted, yet we get more popping and flickering. Having weights at 1.0 gives a good balance of the two conflicting terms.



(a) (c) (d)

Figure 12: *Different Weights. (a) Coherence: 1.0 Texture: 0.5 (b) Coherence: 1.0 Texture: 1.0 (c) Coherence: 0.5 Texture: 1.0. Top Row: Isotropic Synthesis Bottom Row: Anisotropic Synthesis.*

One nice property of our method is that we don't need any special handling for camera zoom. As the camera zooms in, or the object comes close to the camera, the Coherence term of the objective function tries to brow the texture, while the Texture term tries to maintain the same scale/structure of the texture. The result is a perception of getting closer to the object without pattern growing too much. Please watch the accompanying video for the illustration of this effect, as well screenshots in Figure 13.



(a) Frame 1 (c) Frame 20 (d) Frame 40

Figure 13: *Zoom: As we zoom in, each frame resembles our pattern while still satisfying temporal coherence.*

In Figure 16 you can find comparisons of our method with doing standard object space texture mapping. In column (b) are our Isotropic Synthesis results, as you can see, although the texture is warped to follow a given orientation, there is no foreshortening or stretching due to prospective projection, and the pattern spacing remains constant in image space.

The main limitation of the method is that there still can be significant popping and blurring, since in many cases it's not possible for both coherence and texture terms to be satisfied and something has to give.

Some aspects of the method are not as successful as others, for example using custom fill order based on morphological skeleton did

not make much of a difference over using scan-line order. Considering the increased computational cost added by pre-computation of the fill order, it does not seem very practical. Also, modifying the mask during patch sampling to respect occlusion boundaries and creases does not produce visually drastic results. A further investigations are needed to prove or disprove this result.

8 Future Work

Abstracted Flow Motion is a very strong shape cue, and even if the texture lives in image space, and is not foreshortened, if it moves with underlying 3D motion, it will appear 3D. In our future work we hope to introduce ideas of [Breslav et al. 2007] of abstracting flow. Perhaps an additional coherence term, as well expended candidate search space during correction step will allow for the system to find a better solution at expanse of some sliding. We are hoping that additional sliding will allow for the decreased amount of flickering and popping.

Better Tone Correction Our current simple tone correction only works on one layer of synthesis, ideally it should account for multiple layers of texture and media simulation (i.e.optimize blending weights between layers).

Error Maps It could be interesting to try to use per-pixel error values (see Figure 14) to fade out some areas of texture where error is high, or to use this information in multi-layer tone correction. So the error would be used to try to have one layer show up more where there is small overall error and less where there is high error. This could work if for different layers, error occurs in different areas (may not necessary be true).

(a)Texture Error (b) Coherence Error (c) Total Error

Figure 14: *Per-Pixel Error Maps.*

Orientations Introducing new orientations to be used by the method, both automatic and user assisted could be useful. Especially ways to abstract the orientations, and warp them based on human perception to give different looks.

Coherent Stylized Silhouettes Extending the method to handle coherent stylized silhouettes and other feature lines might be an interesting path for future research.

Patch Warping More accurate patch warping might produce better results.

Temporal Coordinate Smoothing In Section 5.3 we described a method for blending coordinated spatially in a way that only blurred pixels at the boundaries. Similarly, perhaps we can blur coordinates temporally, trying to minimize popping and flickering.

Features Evaluations Our current evaluation of the effects of using different features for similarity measure of neighbourhoods in Example Pattern (Section 4.5) needs more evaluation.

Real-Time Version Current system is far from real-time performance, all the calculations are done on the CPU, and can take up to 5 minutes for a single frame to be synthesised. With game industry being larger then animation one, developing a real-time version of the algorithm is an interesting problem.

Video Since our textures live in image space, it's natural to try to apply the method to work with video as an input, however, there are a number of hard problems to be solved, like getting good occlusion and dis-occlusion maps.

9 Acknowledgments

I would like to thank my supervisor, Aaron Hertzmann, for his guidance over the duration of the Masters program, Karan Singh for reading the report and giving feedback, Alexandrina Orzan for her hard work trying to integrate Blender into the pipeline, Martin de Lasa for sharing his Linear Algebra library, and my family, friends, and other DGP Lab Graduate Students for providing valuable feedback and support.

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. *Proceedings of the 2001 symposium on Interactive 3D graphics*, 217–226.

BÉNARD, P., BOUSSEAU, A., AND THOLLOT, J. 2009. Dynamic solid textures for real-time coherent stylization. *Proceedings of the 2009 symposium on Interactive 3D ...* (Jan).

BOUSSEAU, A., KAPLAN, M., THOLLOT, J., AND SILLION, F. 2006. Interactive watercolor rendering with temporal coherence and abstraction. *Proceedings of the 4th international symposium on Non ...* (Jan).

BRESLAV, S., SZERSZEN, K., MARKOSIAN, L., BARLA, P., AND THOLLOT, J. 2007. Dynamic 2d patterns for shading 3d scenes. *SIGGRAPH '07: SIGGRAPH 2007 papers* (Aug).

BREU, H., GIL, J., KIRKPATRICK, D., AND WERMAN, M. 1995. Linear time euclidean distance transform algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence 17*, 5.

CANNY, J. 1987. A computational approach to edge detection. *Readings in computer vision: issues* (Jan).

COCONU, L., DEUSSEN, O., AND HEGE, H. 2006. Real-time pen-and-ink illustration of landscapes. *Proceedings of the 4th international symposium on Non ...* (Jan).

CUNZI, M., THOLLOT, J., PARIS, S., AND DEBUNNE, G. 2003. Dynamic canvas for non-photorealistic walkthroughs. *Graphics Interface 2003* (Jan).

DURAND, F., OSTROMOUKHOV, V., MILLER, M., DURANLEAU, F., AND DORSEY, J. 2001. Decoupling strokes and high-level attributes for interactive traditional drawing. *Proceedings of Eurographics Rendering Workshop'01* (May), 71–82.

EFROS, A., AND FREEMAN, W. 2001. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001* (Jan).

EISENACHER, C., AND LEFEBVRE, S. 2008. Texture synthesis from photographs. *Computer Graphics ...* (Jan).

EISSELE, M., WEISKOPF, D., AND ERTL, T. 2004. Frame-to-frame coherent halftoning in image space. *Theory and Practice of Computer Graphics* (Jan).

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *SIGGRAPH '07: SIGGRAPH 2007 papers* (Aug).

FREUDENBERG, B., MASUCH, M., AND STROTHOTTE, T. 2002. Real-time halftoning: A primitive for non-photorealistic shading. *Proceedings of the 13th Eurographics workshop on ...* (Jan).

GIRSHICK, A., INTERRANTE, V., HAKER, S., AND LEMOINE, T. 2000. Line direction matters: an argument for the use of principal directions in 3d line drawings. *Proceedings of the 1st international symposium on Non- ...* (Jan).

HAN, J., ZHOU, K., WEI, L., GONG, M., BAO, H., AND ZHANG, X. 2006. Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer* (Jan).

HARRIS, C., AND STEPHENS, M. 1988. A combined corner and edge detection. 147–151.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. *International Conference on Computer Graphics and ...* (Jan).

HERTZMANN, A., JACOBS, C., OLIVER, N., AND CURLESS, B. 2001. Image analogies. *International Conference on Computer Graphics and ...* (Jan).

HERTZMANN, A. 2003. A survey of stroke-based rendering. *IEEE COMPUTER GRAPHICS AND APPLICATIONS* (Jan).

KALNINS, R., MARKOSIAN, L., AND MEIER, B. 2002. Wysiwyg npr: Drawing strokes directly on 3d models. *ACM TRANSACTIONS ON GRAPHICS* (Jan).

KALNINS, R., DAVIDSON, P., AND MARKOSIAN, L. 2003. Coherent stylized silhouettes. *ACM Transactions on ...* (Jan).

KAPLAN, M., AND COHEN, E. 2005. A generative model for dynamic canvas motion. *Computational Aesthetics 2005* (Jan).

KIM, Y., YU, J., YU, X., AND LEE, S. 2008. Line-art illustration of dynamic and specular surfaces. *International Conference on Computer Graphics and ...* (Jan).

KLEIN, A. W., LI, W., KAZHDAN, M. M., CORRÊA, W. T., FINKELSTEIN, A., AND FUNKHOUSER, T. A. 2000. Non-photorealistic virtual environments. 527–534.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *International Conference on Computer Graphics and ...* (Jan).

KWATRA, V., ADALSTEINSSON, D., KIM, T., AND KWATRA, N. 2007. Texturing fluids. *IEEE Trans. Visual. Comput. Graphics* (Jan).

LEFEBVRE, S., AND HOPPE, H. 2005. Parallel controllable texture synthesis. *Proceedings of ACM SIGGRAPH 2005* (Jan).

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *Proceedings of ACM SIGGRAPH 2006* (Jan).

MARKOSIAN, L., KOWALSKI, M., AND GOLDSTEIN, D. 1997. Real-time nonphotorealistic rendering. *Proceedings of the 24th annual conference on ...* (Jan).

MEIER, B. 1996. Painterly rendering for animation. *Proceedings of the 23rd annual conference on ...* (Jan).

NARAIN, R., KWATRA, V., LEE, H., KIM, T., CARLSON, M., AND LIN, M. 2007. Feature-guided dynamic texture synthesis on continuous flows. *Eurographics symposium on rendering*.

OSTROMOUKHOV, V., AND HERSCH, R. 1999. Multi-color and artistic dithering. *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (Jul).

PASTOR, O., FREUDENBERG, B., AND STROTHOTTE, T. 2003. Real-time animated stippling. *IEEE Computer Graphics ...* (Jan).

PRAUN, E., HOPPE, H., WEBB, M., AND FINKELSTEIN, A. 2001. Real-time hatching. *Proceedings of the 28th annual conference on Computer ...* (Jan).

SALISBURY, M., WONG, M., HUGHES, J., AND SALESIN, D. 1997. Orientable textures for image-based pen-and-ink illustration. *Proceedings of the 24th annual conference on Computer ...* (Jan).

SNAVELY, N., ZITNICK, C., KANG, S., AND COHEN, M. 2006. Stylizing 2.5-d video. *Proceedings of the 4th international symposium on Non ...* (Jan).

TAPONECCO, F., URNESS, T., AND INTERRANTE, V. 2006. Directional enhancement in texture-based vector field visualization. *Proceedings of the 4th international conference on ...* (Jan).

TURK, G. 2001. Texture synthesis on surfaces. *Proceedings of the 28th annual conference on Computer ...* (Jan).

VANDERHAEGHE, D., BARLA, P., AND THOLLOT, J. 2007. Dynamic point distribution for stroke-based rendering. *... Symposium on Rendering ...* (Jan).

WANG, B., WANG, W., YANG, H., AND SUN, J. 2004. Efficient example-based painting and synthesis of 2d directional texture. *IEEE Transactions on Visualization and Computer ...* (Jan).

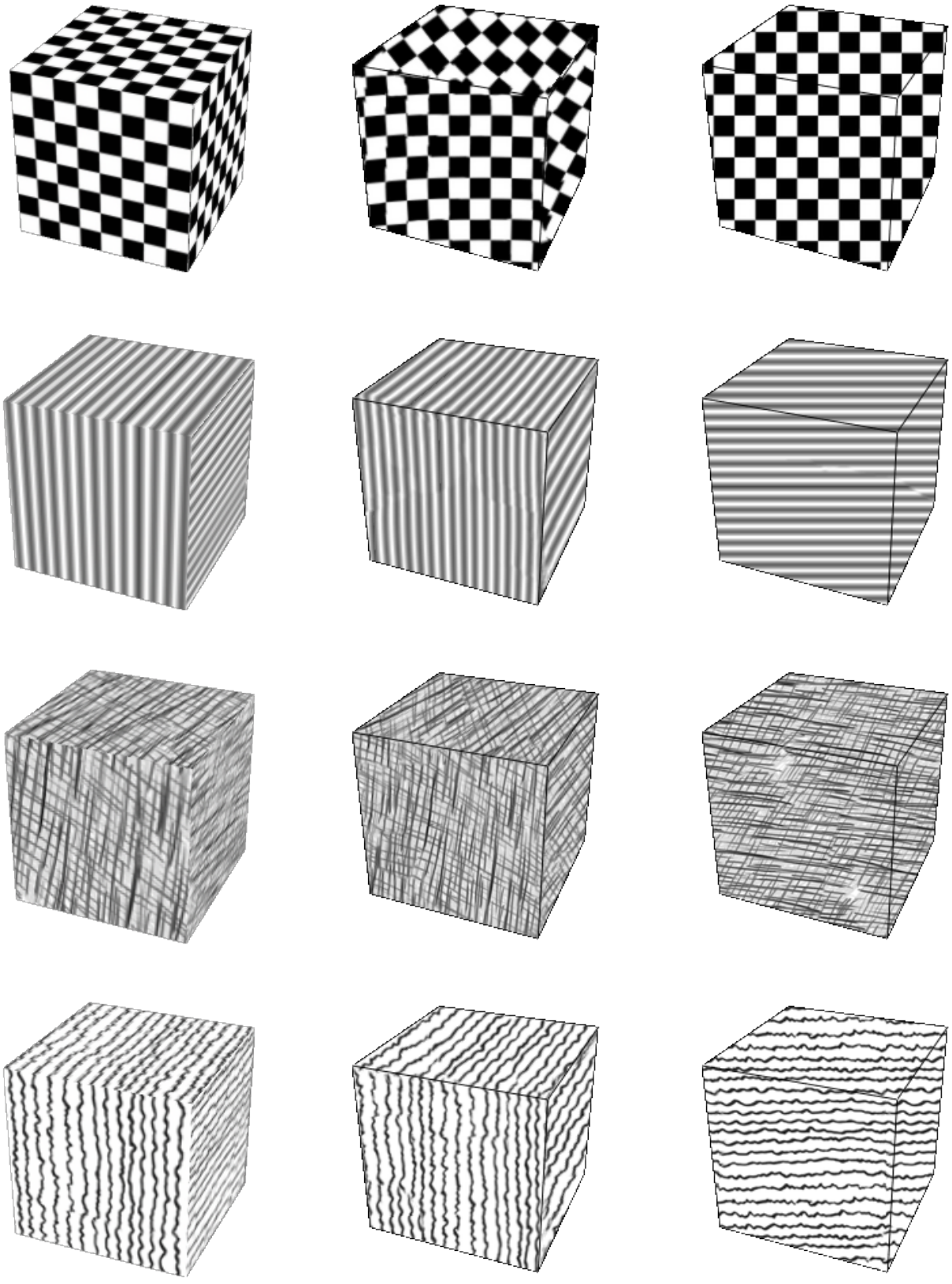
WEI, L., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. *Proceedings of the 28th annual conference on ...* (Jan).

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. *Eurographics Association*.

XU, K., COHEN-OR, D., JU, T., LIU, L., ZHANG, H., ZHOU, S., AND XIONG, Y. 2009. Feature-aligned shape texturing. *ACM Trans. on Graphics (Proceeding of SIGGRAPH Asia 2009)* (Sep).

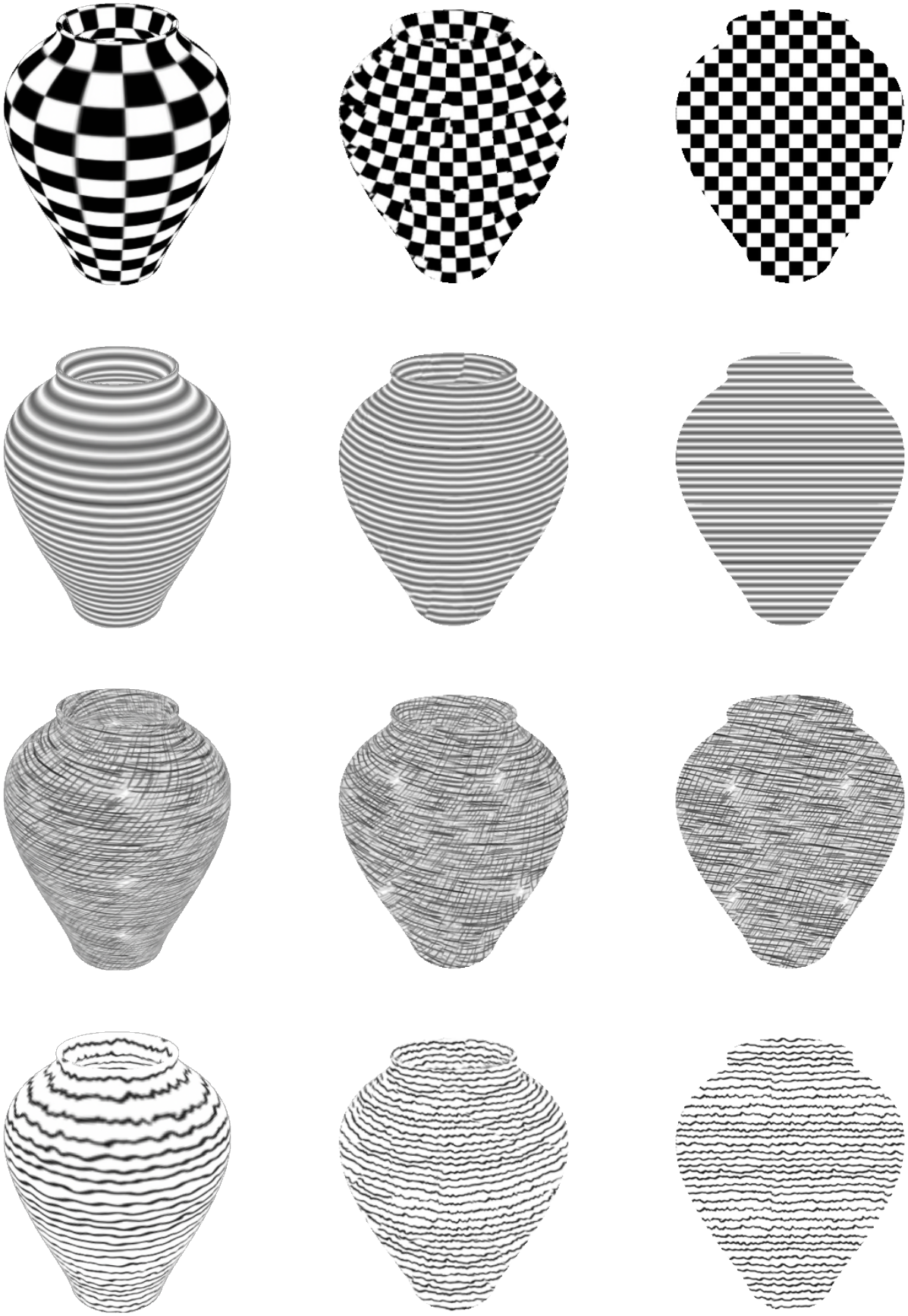
YING, L., HERTZMANN, A., AND BIERMANN, H. 2001. Texture and shape synthesis on surfaces. *... 2001: Proceedings of ...* (Jan).

ZUIDERVELD, K. 1994. Contrast limited adaptive histogram equalization. *Graphics gems IV* (Jan).



(a) (b) (c)

Figure 15: *Object Comparison. (a) Object Space Texture Mapped (b) Image Space Oriented Texture Synthesis(c) Image Space Non-Oriented*



(a) (b) (c)

Figure 16: *Object Comparison. (a) Object Space Texture Mapped (b) Image Space Oriented Texture Synthesis(c) Image Space Non-Oriented*