3

Appearance

3.1 Introduction

Modelling of organ appearance in the virtual scene is the next element of the scenario generation process. As stated in [116], realism of rendered scenes demands complexity, or at least the appearance of complexity. Therefore, we strive to enhance the visual richness of the surface triangle models generated in the previous step by adding further detail. The application of surface textures to geometric models – also referred to as *texturing* – is currently one of the key techniques in interactive computer graphics, to add complexity and thus realism to visual rendering. With this approach, the representation of minute object detail with geometric information can be avoided. Moreover, texture mapping requires only little additional computational effort in the rendering process.

3.1.1 Definitions

The term *texture* does not have a specific general or mathematical definition. Depending on the context, it can have different meanings. For instance, from the point of view of the sense of touch, it denotes the tactile properties of the external surface of an object (e.g., roughness, stickiness). For the gustatory sense, the texture of food relates to specific properties, for instance, crispness or crunchiness. In contrast to this, in material sciences, the term characterizes material properties resulting from the orientation of crystallites.

In the visual domain, one usually thinks of texture as an area which contains a uniform spatial characteristic, which is often attributed to a specific material of the object or surface. In a general sense, the texture appearance is due to the varying light reflectance characteristics of the surface structure. The appearance can be due to different physical effects, ranging from variation of light absorption and reflection (denoted as *flat* textures) caused by submicroscopic structures, via macroscopic effects such as self-shadowing and
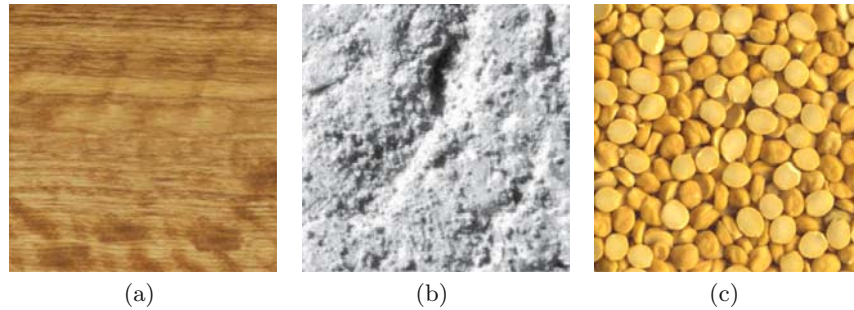
| (a) | (b) | (c) |

Figure 3.1. Different classes of textures. (a) Submircoscopic variation causing spatially varying color perception, (b) Surface relief on macroscopic scale, (c) Collection of objects from distance

-occlusion, to mere collections of objects seen from a distance (Figure 3.1). Texture patterns are usually a combination of all of these classes, and a clear separation is not possible. It should be noted that the appearance of textures is strongly influenced by viewing direction and illumination conditions. These factors are also critical in endoscopic imaging – our source of sample textures – and therefore have to be taken into account.

In addition, several perceptual properties are often attributed to textures, which shall be briefly addressed in the following.

Regularity. Textures can be subdivided into *deterministic* and *stochastic* variants. The former are usually built by repetition of basic patterns in various directions – a typical example being a brick wall. The latter obey a nontrivial probability distribution, and show no dominant frequency or pattern – a typical example being clouds or pasture. Most natural textures fall into the second category.

Isotropy. While isotropic textures are completely independent of direction, and can for instance be freely rotated, anisotropic ones show dominant directions in their appearance – an example of the latter being wood grain.

Resolution. In perceptual terms, texture resolution is related to homogeneity and scale. Usually, textures are invariant under translation, and to a certain degree also to scale. The smallest patch of a texture which exhibits the same perceptual characteristics can be regarded as the texture resolution.

Composition. Textures are often built up in a hierarchical manner containing smaller subtextures. This allows us to treat the different levels of the texture separately. An example of this could be the characteristic substructures of organ surfaces, such as specks on livers.

Many contributions to understanding texture perception in the human visual system, especially discrimination issues, have been made by Bela Julesz
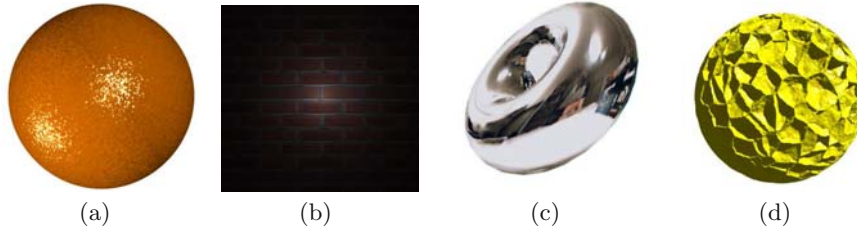
Figure 3.2. Application of texture mapping in computer graphics. (a) Bump mapping, (b) Light map, (c) Environment mapping, (d) Cellular texture

(e.g., [132]). He conjectured that textures with the same power spectrum or with identical second-order correlations cannot easily be discriminated. This topic will be discussed in more detail in Section 3.6.1.

Finally, it should be noted that semantics are often attributed to a specific texture, depending on the viewing context or preknowledge of the beholder. This is for instance true for medical experts, who gain information from an organ's appearance which might not be obvious to laymen.

3.1.2 Texturing in Computer Graphics

The discussion so far has examined textures from a general point of view. However, the application of textures in computer graphics is the main focus of our endeavors, therefore, this area is reviewed with more detail.

The notion of texture mapping was initially introduced in 1974 by Edwin Catmull as a new technique to perform high quality image synthesis [49]. While the major application of texturing is the representation of surface color, several other visual effects based on textures have been developed. In the following some typical techniques are outlined.

Surface color. The standard texturing approach uses one-, two-, or three-dimensional patterns, which are mapped to the surface of an object to determine its color. Initially introduced in [49], this notion is nowadays the most common application of textures. It will also be the major focus of the investigations related to surgical scene generation. Further generalization of the method by including transparency, as described in [94], allows the rendering of a number of natural phenomena, for instance, clouds or smoke.

Illumination enhancement. Real-time rendering algorithms require surface normals to compute light reflection. By using textures to define local perturbations of the normals, an otherwise flat surface appears to have bumps and wrinkles (Fig. 3.2(a)). This method, known as *bump mapping*, was introduced in [28]. A different effect related to lighting can be

achieved with *light maps*. Precomputed illumination data are used to locally modulate light intensity (Fig. 3.2(b)). The increased use of per pixel shading, however, makes this approach more and more obsolete. A different texture-based illumination technique was introduced to represent highly reflective surfaces, which mirror the object surroundings. In order to approximate results obtained from time consuming ray tracing, *environment mapping* can be applied (see e.g., [106]). In this method, a texture built up from images of the surrounding environment is indexed by the directions of rays reflected on the object surface (Fig. 3.2(c)).

Geometry adaptation. An alternative application of texture maps is to define the actual geometric variation of object surfaces. In contrast to bump mapping, this technique creates real bumps in the model. Therefore, high mesh resolutions are required, and interactivity is only possible in limited cases. *Hypertexture* or *cellular texture* methods fall into this category. The former was presented in [207], where object shape was defined as the procedural modification of density in three-dimensional space. The latter was introduced in [84], and denotes the generation of geometric patterns by biologically motivated cellular development. With this technique, surface elements such as scales, feathers, or thorns can be represented (Fig. 3.2(d)).

3.1.3 Relevance to Surgical Education

Apart from the underlying technical aspects, the relevance of appropriate texturing of surgical scenes should also be examined from the medical point of view. Especially in the limited views of minimally-invasive settings, where the surgical site is accessed through natural orifices or small incisions in the skin, texture contains several cues facilitating spatial navigation. As discussed in [32, 196], changes in texture characteristics according to perspective distortion provide knowledge about depth and orientation. Since endoscopic interventions are usually performed with monoscopic cameras, these monocular depth cues contain significant additional semantic information. Apart from this, different kinds of texture of various organs provide further cues for orientation. This is especially true for laparoscopic procedures in the human abdomen, where a large variety of structures is present. Typical organ surface textures allow a surgeon to infer his position from the limited endoscopic view. For instance, fat, intestines, or liver all exhibit characteristic surface patterns and color. Moreover, the variation of individual organ textures hints at possible pathological changes. The distinct appearance of the uterine endometrium could for instance provide knowledge about the presence of endometritis or hyperplasia. Thus, in an advanced education system, the organ textures should also reflect the pathologies present in the training scenario. In terms of the level of realism, appropriately varying textures represent a significant element of a surgical scene. They affect the training of low-level skills, e.g., orientation in the surgical site; as well as high-level skills, e.g., decision making according to semantic information.
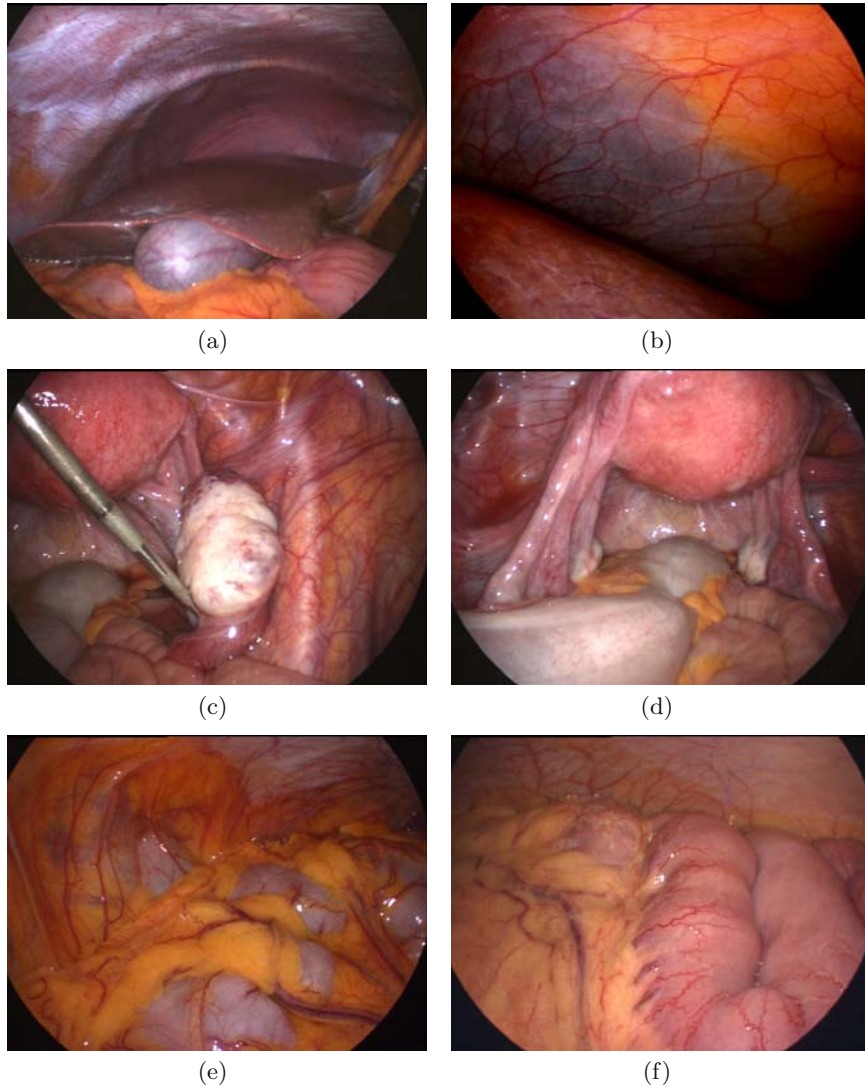
Figure 3.3. Pictures taken during laparoscopic interventions. (a) Cranial section of abdomen showing liver, gallbladder, peritoneum, and diaphragm, (b) Close-up of diaphragm with typical vascular structures, (c) Kaudal section of abdomen depicting uterus, ovary, and peritoneum, (d) Uterus structure, including corpus, fallopian tubes, and other adnexa, (e) Patches of fat covering intestinal structures, (f) Small intestine with fat and typical vessel trees

As discussed earlier, the methods presented in the following sections of this chapter were developed for two settings – gynecological laparoscopy and hysteroscopy. Because of different access methods to the uterus, varying
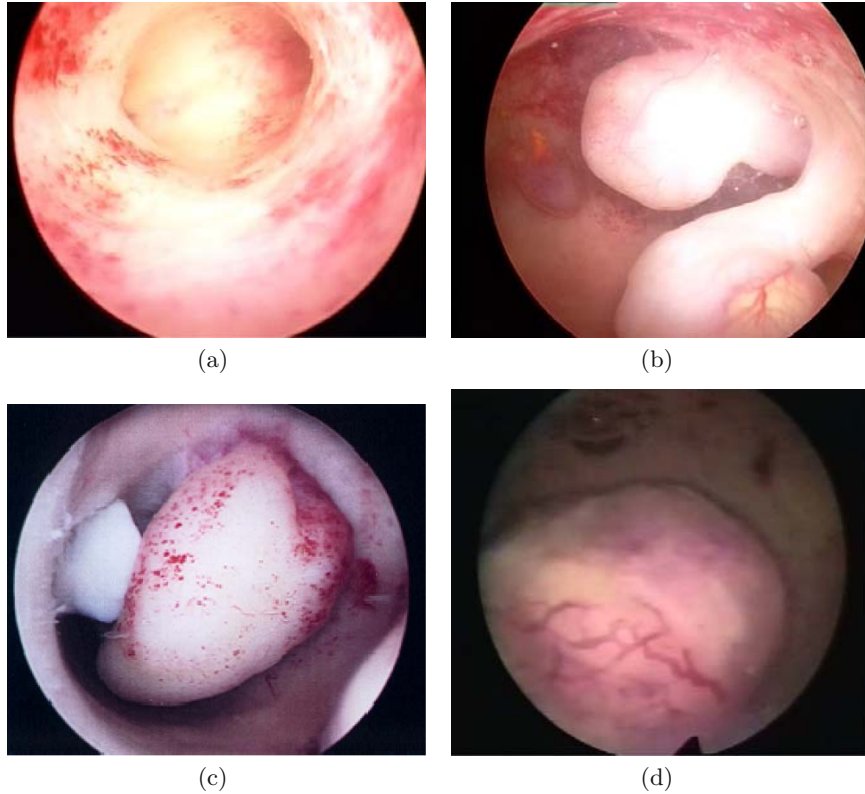
Figure 3.4. Pictures taken during hysteroscopies. (a) View inside the fluid-filled uterine cavity with no abnormal findings, (b) Small polyps growing from the endometrial wall, (c) Larger polyp inside uterine cavity, (d) Uterine leiomyoma with typical vessel structures

requirementsfor organ texturing have to be met. In Figures 3.3 and 3.4, snapshots of real interventions show characteristic views of both approaches. Typical surface textures can be seen in both collections. It should be noted that the imaging conditions vary greatly for the two methods. While in the gas-insuflated abdomen high quality pictures of all structures can easily be taken, the intrauterine image quality is usually quite low. The depicted selection represents samples taken under nearly optimal conditions.

3.1.4 Process Elements

In the next sections, all the steps necessary for generation of organ textures will be discussed in detail. The first element is the intraoperative image acquisition to form a ground truth, which is necessary for most of the texture synthesis methods. Then image enhancements are carried out to deal with
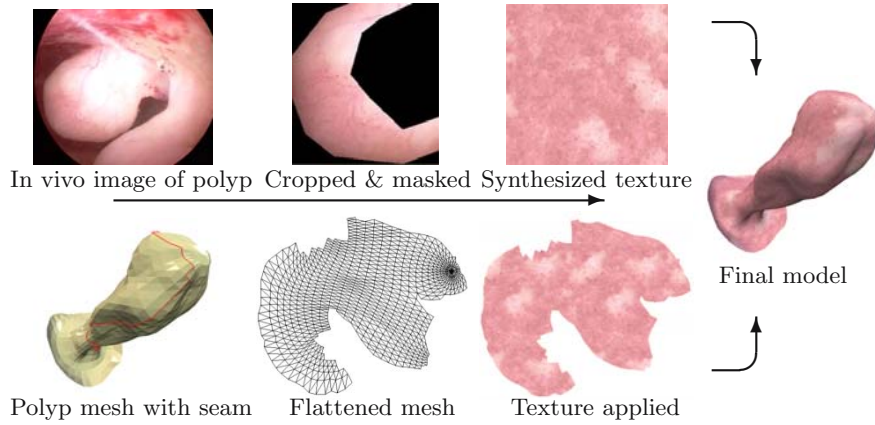
Figure 3.5. Texturing process for hysteroscopy simulation (from [203])

the often suboptimal imaging conditions. Subsequently, the image data can be used for textures synthesis. Thereafter, additional procedural texturing methods can be applied to include further detail in the textures. The final step comprises the mapping of the texture to the geometric organ surface.

It should be noted that tissue is often covered by vascular structures. These vessel patterns contain vital information for the surgeon, and need to be reproduced appropriately. However, the structures are usually too complex to be synthesized with texture generation approaches. Moreover, vessels also have to be included in the simulation as geometric objects, since they will be sources of bleeding when cut. Therefore, vascular trees have to be treated separately from the underlying *base texture*. The discussion in this chapter is constrained to the generation of varying, organ-specific base textures. The integration of vessel systems will be briefly reviewed in Chapter 5. Nevertheless, it should be noted that some surface structures exist – for instance, specks on livers, or follicles on ovaries – which will be addressed in this chapter.

In order to provide an overview of all necessary steps, the texturing approach taken for the hysteroscopy setting is depicted in Figure 3.5. The method uses tileable, variable textures created from in vivo images for the texture synthesis. Textures are mapped to 3D mesh geometry by mesh parameterization, taking into account the visibility of seams and distortion reduction. More details of these process elements, as well as those for laparoscopic textures, are explained after a review of previous work.

3.2 Previous Approaches in Surgery Simulation

This section focuses on a survey of the methods used for generation of textures in the context of surgical simulation or for other related natural phenomena.
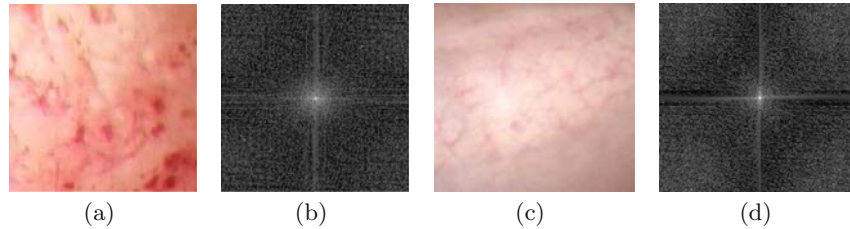
(a)   (b)   (c)   (d)

Figure 3.6. Fourier spectra of different organ textures. (a) Endometrium texture t_1, (b) Magnitude of Fourier transform $log(|T_1|)$, (c) Bowel texture t_2, (d) Magnitude of Fourier transform $log(|T_2|)$

More general work in texture generation and mapping will be addressed below. More precisely, procedural texturing will be detailed in Section 3.5, human texture perception in Section 3.6.1, and a review of pixel- and patch-based texture synthesis will be provided in Section 3.6.2. Nevertheless, here previous attempts related to surgical simulation are examined, in order to put the presented texturing methods into an application-oriented context.

The most basic approach for creating organ textures suggested in the past is direct texture painting. This usually requires a medical illustrator to manually draw the texture with the help of appropriate tools. In [233] a two-dimensional painting method was used to obtain textures for a virtual environment targeting biomechanical analysis and education. The illustrator has to ensure in a trial and error process that the 2D drawings map to the 3D geometry without distortion. More advanced approaches proposed to draw textures directly in 3D. Initial steps in this direction were reported in [113]; however, the lack of a true 3D user interface makes difficult an intuitive usage of the system. Improved tools, for instance using haptic interfaces, were presented later, e.g., in [2, 130, 20]. Nevertheless, the direct painting of textures does not meet the requirements of variable training scene generation, since a new texture would have to be created or adapted for every new organ. While high texture detail might be achievable with such an approach, its costly and time consuming nature prohibit its usage in our setting.

An interesting alternative to direct painting in the spatial domain is presented in [160], where the author suggests drawing a texture in the frequency domain. It is possible to create patterns found in wood or canvas with this method. Nevertheless, it is admitted that the correct spectrum for natural textures might be as complicated as the texture itself. In Figure 3.6, logarithmic transformations of the Fourier spectra of two representative samples are depicted. Unfortunately, no patterns can be identified which would encourage closer examination of texture drawing in the frequency domain.

In [154] an approach using polyhedron decomposition is described which allows the treatment of each surface triangle as an independent entity with its color information stored in a unique texture space. The triangles are ar-

ranged in the parameter space to optimize area coverage. Textures representing anatomical details are manually placed by medical experts with a 3D tool, and interactively mapped to the surface. Blending between different texture patches reduces discontinuities. The method has been applied in a simulator for minimally invasive neurosurgery. Real images acquired during endoscopy are used to manually generate the texture patches. Again, these methods require too much manual input from the user.

A different strategy using texture samples and avoiding distortion, discontinuity, and repetitiveness is presented in [194]. It is based on the triangular tiling of a surface at a user-specified scale. The tiling defines a local parameterization, which is used to map triangular texture patches to the surface triangles. If the tiling is relatively homogeneous, distortions are expected to remain favorable. Due to the selected tiling, texture patches have to match their color and local derivative at the borders. The patches are either created manually or semiautomatically. In the former case, the patches are hand drawn samples, or manually edited real images. In the latter case, the textures are synthesized based on nonperiodic cellular patterns [297] or fractal noise [208]. This method was used to apply texture samples to a 3D mesh of a liver. The patches were synthesized with Worley's approach, and four sets of equilateral triangles with isotropic homogeneous texture were applied. This work was later extended in [193] by adding dynamic effects directly to the surface texture, for instance, cauterization marks or blood drops. These effects were added with direct painting paradigms, and distortions were avoided by considering the local Jacobians. The texture variation achieved with these techniques is rather limited, due to the small number of triangular patches.

A different approach for obtaining textures for medical scenes is the mapping of real volumetric data to surfaces. In [224], the Visible Human data set is utilized to obtain texture for organs. The authors segment individual structures from the data. Thereafter, polygon models are placed along the segmented boundaries and textured directly from the volumetric data. New textured surfaces, created for instance during cutting procedures, can also be obtained on the fly. Unfortunately, the segmentation of the anatomical structures in the dataset is time-intensive and tedious, and can be carried out only with limited precision. Moreover, variable scenes cannot be generated with this approach. Similarly, volumetric texture mapping is also proposed in [162]. A tetrahedral mesh is mapped to the Visible Human data, which can be cut and deformed. An interesting alternative has been suggested in [144]; namely to generate organ-specific CT color lookup tables based on the Visible Human CT and cryosectional data. By the mapping of Hounsfield units to RGB values, different CT datasets can be colorized. Nevertheless, this method provides only an approximate colorization of organs, and does not include detailed surface information. Moreover, the appearance of specific pathologies cannot be obtained with this approach.

Another recent trend in surgical simulation is the use of methods inspired by image-based rendering [136] for realistic visualization. For instance, in [82],

real images are combined with artificial specular reflections, modulated by a set of reflectance maps, to simulate different views by the endoscopic camera. The maps are generated with Perlin fractal noise. As an application example, this method was used in bronchioscopy simulation. The model geometries are obtained from CT images, while the textures are acquired from live endoscopic images, and later registered to the model. Unfortunately, the authors do not discuss the texturing process in more detail. It is not clear how correct correspondences between texture and geometry are obtained. Moreover, it is not explained how problems like missing regions or highlights visible in the actual endoscopic images are handled. Finally, the approach would have to be repeated for every new model. A similar approach is reported in [218], where specular highlights are created with environment mapping, utilizing current graphics hardware extensions. However, the authors do not report on how the cube map of the texture is generated in order to avoid distortions. Recently, a combination of conventional texture mapping, image mosaicking, view-dependent texture mapping, and light source shading has been proposed in [129] for realistic rendering of surgical scenes. The authors assume that the background in endoscopic simulation does not change significantly, thus enabling the rendering of novel views with image-based rendering principles. First, mosaicking is applied to composite background images from an endoscopic video sequence. Since different viewing angles cause varying specular highlights, a number of these background images are created for different angles. During runtime, view-dependent blending of these images is performed. However, the scene foreground, i.e., the actual interactive liver model, is rendered with standard methods. Therefore, a large difference in rendering quality between foreground and background can be noticed in the scene.

For completeness, reaction-diffusion processes, which have been suggested for the synthesis of natural textures [295, 270], should also be mentioned. With this approach, chemical processes describing pattern formation in biological morphogenesis are simulated. As an example, typical animal fur patterns can be reproduced with this method. However, there is no straightforward extension of reaction-diffusion texture generation to the formation of specific organ patterns.
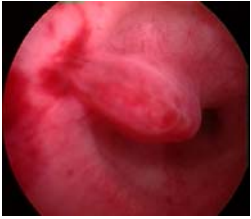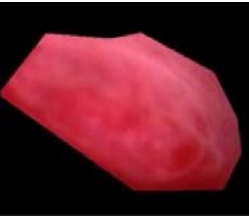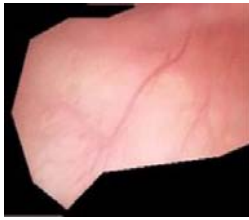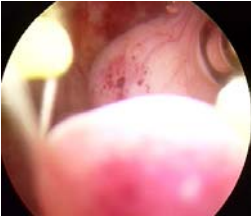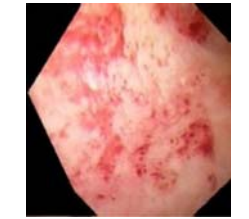
3.3 Data Acquisition and Enhancement

3.3.1 In Vivo Image Acquisition

The first step in the texture generation chain is the acquisition of image data from real interventions. This is necessary, since for the majority of organs an analysis/synthesis paradigm is followed. Only in limited simple cases can a texture be defined with a purely functional approach. Therefore, a ground truth covering the visual appearance of organs in healthy as well as pathologic conditions has to be obtained. Due to the varying boundary conditions

of laparoscopic and hysteroscopic interventions, different image acquisition techniques have to be used.

Table 3.1. Excerpt of in vivo image database

Original image	Cropped and masked image	Properties
		• ID: 105 • Object: polyp • Age: postmenopausal • Resolution: medium • Size: 374x312 • Focus: excellent
		• ID: 142 • Object: polyp • Age: unknown • Resolution: medium • Size: 198x140 • Focus: good
		• ID: 149 • Object: polyp • Age: unknown • Resolution: medium • Size: 253x111 • Focus: good
		• ID: 151 • Object: endometrium • Age: unknown • Resolution: high • Size: 329x292 • Focus: good

The largest limitations are encountered in hysteroscopy, mainly due to time constraints and poor visibility inside the uterus. The former are due to the problem of fluid overload caused by absorption or intravasation. Operation

time and the consumed quantity of the distension media have to be closely monitored. Generally, an operation time of 30 mins should not be exceeded. Due to this, only a limited amount of time can be spent obtaining high quality images of structures inside the uterus. Moreover, the view usually is obscured by floating tissue fragments, blood, and endometrial fibers. Therefore, instead of acquiring specific high-quality images, pictures were obtained from 10 hours of in vivo video taken during various hysteroscopies. Images representing different structures in the uterine cavity were selected, labeled, and input into a texture database. Usually, high resolution views of characteristic tissues were chosen, which optimally were perpendicular to the surface and without strong camera spotlight effects. 69 different samples were obtained that contained usable textures of various types of tissue. In Table 3.1 a few categorized textures of the searchable database are visualized.

In contrast to this, the acquisition process during laparoscopy is less complex. Since time constraints are less severe, and individual organs can be selectively brought into view, higher quality samples can be taken. In order to minimize the disturbance of the normal surgical procedure, snapshots can be taken by pressing a foot pedal. With this approach, about 400 high resolution true color images were obtained from 16 different laparoscopies.

3.3.2 Image Enhancement

Before the acquired images can be used for texture synthesis, a few additional preprocessing steps have to be carried out. The first step is to crop from the image the region which contains the appropriate sample for a specific organ texture. It should be noted that the quality of the image can be diminished by a number of effects. Texture samples can show a trend due to orientation, scale, or viewing angle. This is especially the case for highly curved organ surfaces. In order to reduce this trend, images should be taken from flat areas of an organ with a perpendicular viewing direction. However, perspective distortion present in endoscopic cameras cannot be avoided. A previous calibration of the optical system before image acquisition might reduce these effects; however, such a process has to be performed before every operation, and cannot easily be integrated into the normal workflow. Moreover, image quality might be further deteriorated due to blur – a particular obstacle in hysteroscopy. Finally, problems can be encountered due to the interlaced imaging mode, in which half images are acquired alternatingly for even and odd lines. This sometimes adds high frequency noise to the data.

The cropped samples obtained in the first step should contain relatively homogeneous regions of the desired base texture. However, additional structures might be present in the sample, which have to be removed before further analysis. This includes highlights coming from the camera system. Since a perpendicular viewing direction is sought, these interfering highlights are usually part of the image. In laparoscopy, specular reflections are present; while in hysteroscopy, diffuse intensity shifts are encountered. Apart from this, other
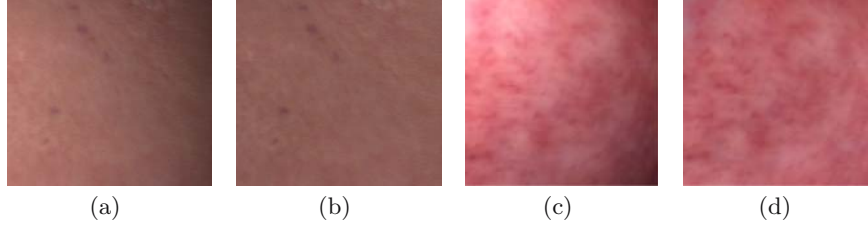
(a)   (b)   (c)   (d)

Figure 3.7. Reduction of intensity drift for texture samples. (a) Texture sample of liver surface, (b) Drift reduction with quadratic fit, (c) Texture sample of uterus surface, (d) Drift reduction with quadratic fit

surface structures which should not be part of the base texture also have to be removed, e.g., vessels, or structures like follicles on ovaries. Therefore, regions containing these types of structures are masked in the images. It should be noted that this results in arbitrarily shaped texture samples with masked pixels which have to be taken into account in the later processing steps.

Illumination trends due to drifts in intensity and saturation caused by unhomogeneous distribution of light energy and diffuse reflection on curved organ surfaces are another difficulty. While these effects might not be directly noticeable by a human observer, they have to be rectified before the synthesis process. One approach to alleviate this situation is to perform a filtering in HSV space based on least squares fitting. To this end, the RGB color image is converted to HSV space.

$$H = \begin{cases} 60 \cdot \dfrac{G-B}{Max-Min}, & \text{if} \quad Max = R \\ 60 \cdot \dfrac{B-R}{Max-Min} + 120, & \text{if} \quad Max = G \\ 60 \cdot \dfrac{R-G}{Max-Min} + 240, & \text{if} \quad Max = B \end{cases} \qquad (3.1)$$

$$S = 1 - \frac{Min}{Max}$$

$$V = Max\,,$$

where Max is the maximum of the color channel triplet $(R, G, B) \in [0.0, 1.0]^3$, and Min the minimum of these values (with $Max - Min \neq 0$ and $Max \neq 0$). In HSV space, drift will be corrected by separately processing the Hue (H) and Value (V) channel. A least squares approximation is performed with a quadratic or cubic fit function.

$$f(x, y) = \sum_{n=0}^{k} \sum_{m=0}^{k} c_{n,m}\, x^n y^m, \quad \text{with} \quad k \in \{2, 3\}\,. \qquad (3.2)$$

In order to determine the unknown coefficients $c_{n,m}$, the least squares solution to an overdetermined system of linear equations has to be obtained. In the case of a quadratic fit, the equation system takes the form

$$
\begin{bmatrix}
1 & x_0 & y_0 & x_0^2 & x_0\,y_0 & y_0^2 \\
1 & x_1 & y_1 & x_1^2 & x_1\,y_1 & y_1^2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & x_i & y_i & x_i^2 & x_i\,y_i & y_i^2
\end{bmatrix}
\cdot
\begin{pmatrix}
c_{0,0} \\
c_{1,0} \\
c_{0,1} \\
c_{2,0} \\
c_{1,1} \\
c_{0,2}
\end{pmatrix}
=
\begin{pmatrix}
v_0 \\
v_1 \\
\vdots \\
v_i
\end{pmatrix},
\tag{3.3}
$$

where v_i are the respective H or V values at positions (x_i, y_i) of the i unmasked texels in the sample texture. Equation 3.3 can be solved with standard solvers, e.g., Householder reflections or Givens rotations. After the coefficients of the fit function have been determined, the drift of the initial sample can be adjusted by subtracting the quadratic or cubic fit, and subsequently adding the mean of the sample texels. Finally, the new values are converted back to RGB color space (examples of the overall process are shown in Figure 3.7).

$$
H_i = \left\lfloor \frac{H}{60} \right\rfloor \bmod 6, \; f = \frac{H}{60} - H_i,
\tag{3.4}
$$

$$
p = V \cdot (1 - S), \; q = V \cdot \big(1 - (S \cdot f)\big), \; t = V \cdot \Big(1 - \big(S \cdot (1 - f)\big)\Big).
$$

$$
\begin{aligned}
R &= V, & G &= t, & B &= p, & \text{if} \quad H_i &= 0 \\
R &= q, & G &= V, & B &= p, & \text{if} \quad H_i &= 1 \\
R &= p, & G &= V, & B &= t, & \text{if} \quad H_i &= 2 \\
R &= p, & G &= q, & B &= V, & \text{if} \quad H_i &= 3 \\
R &= t, & G &= p, & B &= V, & \text{if} \quad H_i &= 4 \\
R &= V, & G &= p, & B &= q, & \text{if} \quad H_i &= 5
\end{aligned}
$$

The final step in the image preprocessing is the reduction of colors by quantization. The acquired true color images can in principle contain more than 16 million different color values. As we will see later, computation time is often directly related to the number of colors. Therefore, it is advisable to limit the amount of the latter. The sample images are well suited for this step, since only a limited number of tones and intensities is present. Moreover, errors created during the process are obscured by the stochastic nature of the samples. Instead of uniform quantization of the color channels, more sophisticated algorithms such as the ones discussed in [99] or [298] should be used. Typically, 32 colors are sufficient for the majority of the samples. Figure 3.8 shows examples of the color reduction.
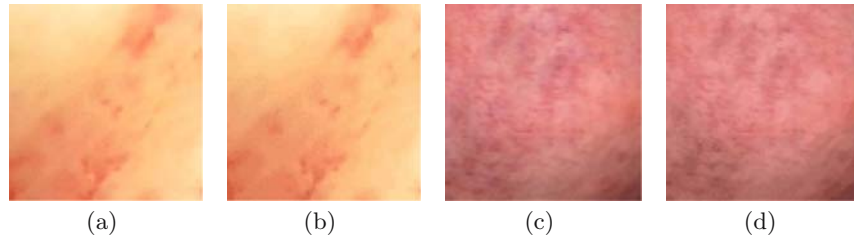
<center>(a)   (b)   (c)   (d)</center>

Figure 3.8. Color reduction by quantization. (a) Original texture sample of endometrial wall, (b) Quantized sample with 32 colors, (c) Original texture sample of uterus, (d) Quantized sample with 32 colors

3.4 Base Texture Generation

The next step in the texturing chain is the generation of the base textures. Generally, two different strategies could be followed for this: empirical procedural texture generation based on closed mathematical formulations, or analytical approaches applying example-based synthesis. Mainly the latter are used in the scenario definition process, since procedural methods suffer from several limitations. The main drawback is the nonintuitive control of the algorithms. First, an appropriate mathematical representation has to be found that describes a specific pattern. Moreover, a number of parameters have to be manually supplied to control the results. Nevertheless, a few simpler textures can be generated with the former strategy. Therefore we first discuss these methods, before moving on to the other category.

Furthermore, it should be noted that the dimensionality of the generation has to be considered. Directly obtaining 3D textures avoids several texturing problems encountered in 2D approaches. The most significant advantage is the circumvention of the surface mapping step. Therefore, inherent obstacles in 2D, such as distortion or continuity problems, can be avoided. Instead, the textured object is "carved" from the block of texture. This works especially well for patterns that emerge from real 3D phenomena, e.g., wood or marble. However, 3D strategies perform well only with relatively homogeneous and isotropic textures. Moreover, it should be noted that not all characteristics of some specific 2D patterns can easily be reproduced with a 3D texture. Consider as a typical example a texture with uniform, filled circles. A volumetric version should contain uniformly distributed spheres; however, an arbitrary surface cut from this block would not contain circles of uniform size. In this sense, 2D approaches appear to be more flexible. Nevertheless, in this case an appropriate solution for mapping the texture to a 3D surface has to be found. Both 2D and 3D textures have been used in texture generation, and thus both will be discussed in the following.

3.5 Procedural Textures

3.5.1 Review

The notion of procedural texturing was initially coined in [208]. Instead of explicitly specifying and storing all details of a texture, they are abstracted into a function or algorithm (i.e., the procedure) which can then be evaluated on the fly. With this approach, problems with texture memory or bandwidth limitations can be avoided. Moreover, due to the procedural design, texture resolution is infinite, since magnifications can be interactively generated. Another concept which has to be mentioned in this context is solid texturing. Introduced by Ken Perlin at the same time as [94] and [206], this paradigm decouples texture from geometry and defines it in 3D space. This obviates the mapping of a 2D texture to a 3D surface, thus also avoiding any distortions or discontinuities.

The first usage of a mathematical function to define a solid texture can be found in [94]. Inspired by [229], the author uses Fourier expansions to represent real-world detail at a statistical level, for instance to represent clouds or trees.

$$T(x_s, y_s, z_s) = \sum_{i=1}^{n} c_i \frac{\big(\sin(\omega_i x_s) + P_{x_i} + 1\big)}{2} \cdot \sum_{i=1}^{n} c_i \frac{\big(\sin(\omega_i y_s) + P_{y_i} + 1\big)}{2}$$

(3.5)

where P_{x_i} and P_{y_i} represent phase shift functions which are added to avoid regularities in the pattern. c_i and ω_i provide some control over the amplitude and period of the signal.

Also in [206], basic solid texturing was applied. It is suggested that projection functions be defined to extrude 2D texture into 3D space – in the simplest case this can be an orthogonal projection, for instance, to generate 3D wood textures from concentric circles. Moreover, the usage of combination functions to merge different solid textures is also described. A typical granite texture could for example be generated by combining three basic solid textures.

The most sophisticated procedural approach has been described in [208]. It is based on the application of pseudo-noise, which is used as a numerical representation of the randomness found in natural structures. It should be noted that pseudo-noise only gives the appearance of randomness; nevertheless, it is sufficient for generating convincing textures. The core element of the algorithm is the definition of a random noise function $\nu : I\!R^n \longrightarrow I\!R$, which should be band-limited. Concentrating the energy in a small section of the frequency spectrum results in similar size and anisotropy of the apparently random variations. Moreover, the outcome should be statistically invariant under translation and rotation. Since we aim at generating solid textures, the following discussion is limited to defining ν for $n = 3$.

The first step is to define a grid of pseudo-random gradient vectors $\mathbf{g}$ positioned at 3D lattice points $\mathbf{q}$. These random vectors are generated by a
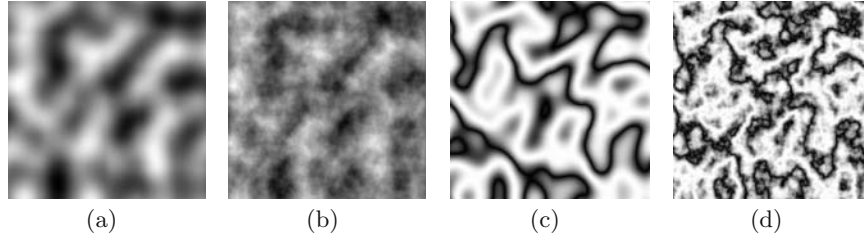
(a)   (b)   (c)   (d)

Figure 3.9. Examples of Perlin procedural textures. (a) Fractal turbulence τ with $m = 0$, (b) Fractal turbulence τ with $m = 3$, (c) Marble texture μ with $m = 0$, (d) Marble texture μ with $m = 4$

Monte Carlo simulation, which computes N points uniformly distributed on the surface of a unit sphere. The vectors are stored in an array G, which is usually periodic to limit the number of vectors. This is acceptable as long as the repetition occurs over a long distance – in practice $N = 256$ yields good results. In the next step, vectors $\mathbf{g}$ are randomly assigned to lattice points. This can be done with a heuristic indexing function based on lattice coordinates i, j, k.

$$\mathbf{g}(i, j, k) = G\Big[\tau\big[(i + \tau[(j + \tau[k \mod N]) \mod N]) \mod N\big]\Big], \quad (3.6)$$

where τ represents an array containing a precomputed random permutation of the numbers 0 to $N - 1$. After assigning the random gradient vectors using Equation 3.6, the noise function can be evaluated at an arbitrary 3D position $\mathbf{p} = (x, y, z)$ by calculating a weighted sum of inner products at the eight closest lattice points $\mathbf{q}_{i,j,k}$.

$$\nu(x, y, z) = \sum_{i=\lfloor x \rfloor}^{\lfloor x \rfloor + 1} \sum_{j=\lfloor y \rfloor}^{\lfloor y \rfloor + 1} \sum_{k=\lfloor z \rfloor}^{\lfloor z \rfloor + 1} \omega(\mathbf{p} - \mathbf{q}_{i,j,k}) \big[\mathbf{g}(i, j, k) \cdot (\mathbf{p} - \mathbf{q}_{i,j,k})\big] . \quad (3.7)$$

Here, ω represents an S-shaped, cubic drop-off filter to weigh the interpolants in each dimension.

The resulting noise function can now be used to generate textures with random appearance, either by direct mapping to colors, or as an additional parameter in a composite mathematical description. One typical approach is to generate fractal turbulence by iterating over octaves.

$$\tau(\mathbf{p}) = \sum_{i=0}^{m} \frac{\nu(2^i \cdot \mathbf{p})}{2^i} . \quad (3.8)$$

It should be noted that usually after a few harmonics the changes are less than the resolution of the texture, both in terms of spatial and grey value resolution.

Textures resembling marble can for instance be generated with

$$\mu(\mathbf{p}) = \left|cos(c_1\mathbf{p}_1 + c_2\mathbf{p}_2 + c_3\,\tau(\mathbf{p})\right|, \tag{3.9}$$

where c_1 and c_2 allow control of the period of the typical marble patterns in two main directions, and c_3 denotes the strength of the turbulence. $\mathbf{p}_i$ represents the i-th component of the vector. Examples of procedural texture according to the discussed algorithm are depicted in Figure 3.9.

While Perlin's noise function provides reasonable results, there are a few shortcomings. One problem is the definition of the function, which causes it to evaluate to zero at the lattice points. This might be noticeable as a disturbing regular appearance. Since in general any noise function which fulfills the requirements stated above could be used, other approaches for procedural noise generation were also scrutinized.

In [159] a sparse convolution algorithm for three-dimensional noise is introduced. It is synthesized by the convolution of a kernel $h(\mathbf{p})$ with a Poisson noise process of impulses, which are uncorrelated in intensity and distributed at uncorrelated locations in space. Due to the impulse nature of the noise, the convolution reduces to a summation over the impulses.

$$\tilde{\nu}(\mathbf{p}) = \sum_{i=0}^{m} \alpha_i \cdot h(\mathbf{p} - \mathbf{p}_i), \tag{3.10}$$

where $\mathbf{p}_i$ is the location of the i-th impulse and α_i a scaling parameter. Using this in the more general form of Equation 3.8, a new turbulence function is obtained:

$$\tilde{\tau}(\mathbf{p}) = \sum_{i=0}^{m} \beta_i(\nu(\gamma_i\mathbf{p})). \tag{3.11}$$

The parameters β_i and γ_i can be used to individually control the amplitude and scale of all noise basis functions, respectively. As an example for the kernel h, a smooth cosine function was used to obtain a wood-like appearance.

3.5.2 Generation of Simple Textures

While the schemes discussed above are not sufficient to generate the more complex surface patterns found on most organs, they can be used to generate some of the simpler base textures, for instance those of fat or intestine. This can be accomplished by using the noise scalars to linearly interpolate between two typical colors in RGB space. These colors can be selected to be the minimal and maximal color value of a specific source image. In Figure 3.10, two generated example textures applied to artificially generated surfaces are depicted. To assess the quality of this result, the same objects are also shown integrated into a real in vivo scene. The noise was generated with Equation 3.11, using $m = 1, \beta_0 = 5.0, \beta_1 = 10.0, \gamma_0 = 1.0, \gamma_1 = 0.707$.
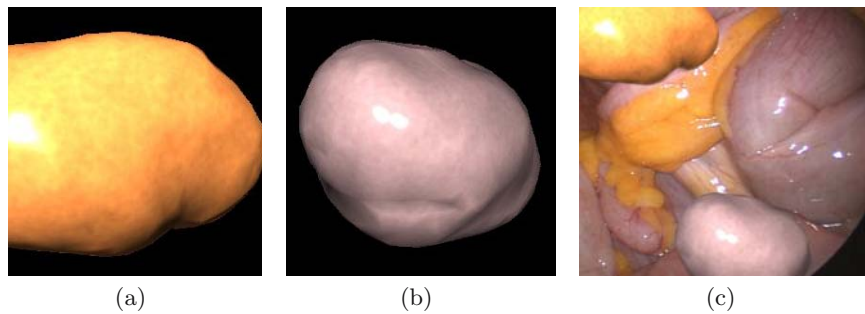
(a)   (b)   (c)

Figure 3.10. Simple textures obtained from procedural approaches. (a) Example texture resembling fat, (b) Example texture resembling bowel, (c) Sample objects integrated into in vivo image

It can be concluded that the presented approach enables the procedural generation of acceptable simple base textures. However, a number of problems have to be solved, which make the approach infeasible in most cases. Manual tuning is required to set the parameters of the noise functions, as well as specifying the correct RGB values to use for interpolation. In addition, color interpolation in RGB space can only provide meaningful output if the sample colors are relatively close to each other. Carrying out the interpolation in HSV space might be more appropriate for this. Moreover, a higher order interpolation with more sampling points might also provide better color variation.

In the context of the scene generation process, procedural texturing also does not fulfill the posed requirements. Synthesis times can be long at higher resolutions, and due to the empirical nature of the approach, several attempts might be necessary to obtain the desired results. Since the scene definition should be carried out by medical experts with the provided tools, this algorithm behavior is not acceptable. Due to the fact that also textures with higher variability should be generated, we focus mainly on texture synthesis by way of example.

3.6 Texture Synthesis

The underlying idea of approaches falling into the synthesis category is texture creation from example. A sample is supplied to an algorithm and analyzed, and then a new texture is automatically created resembling the original one. Different strategies can be used to ensure the similarity between the input and output texture. This paradigm fits quite well into the training scene generation process, since the methods can easily be controlled by novice users, i.e., the medical personnel in our case. The overall process should require as little user intervention as possible. However, if interaction should become necessary or

is desired by the user, control of the method should be intuitive. Interactive control over the placement of texture by the user should also be allowed. Nevertheless, overall synthesis quality should not be limited by providing this option. These boundary conditions will have some influence on the selection of the synthesis algorithm.

3.6.1 Human Texture Perception

A number of texture synthesis approaches are inspired by research examining the human perception of texture. This area of investigation was pioneered by the work described in [132]. Two random visual patterns, either side-by-side, or one contained in the other, were presented to test subjects. The patterns were identical in their probability distribution of the n-th order, but differed in their $(n + 1)$-th-order distribution. The test subjects had to perform a spontaneous visual discrimination task with these patterns. It was observed that textures differing in their first-order statistics can easily be discriminated, while patterns with the same first- and second-order statistics could not be distinguished. This led to the well known conjecture that textures appear indistinguishable to humans if their first- and second order statistics are identical [133].

This assumption, however, was later disproved by the same group [45, 134]. They were able to create textures with iso-second-order patterns which were distinguishable in pre-attentive human visual perception experiments. Nevertheless, inspired by the earlier work, pixel pair statistics are still very common in texture synthesis approaches. Especially for the less structured textures found on human organs, methods based on second-order distributions using pixel pairs yield good results.

More recent work based on psychophysical and neurophysiological examinations indicates that a multichannel spatial frequency and orientation analysis is performed within the visual cortex (see, e.g., [122, 179, 273, 64, 23, 178]). It was suggested that the response of cells in the primary visual cortex of mammals can be modeled with Gabor filter kernels [91]. Thus, similar textures should produce the same responses to a bank of oriented filters.

3.6.2 Review

Texture synthesis methods proposed in the past originate from diverse underlying ideas and follow different strategies. A distinction of the approaches is not straightforward, since often a combination of methods is used. One possible categorization discriminates between pixel- and patch-oriented synthesis. The former synthesizes a pixel at a time, while the latter pastes complete patches into the new texture.

Pixel-Based Synthesis

Synthesis algorithms focusing on single pixels can be further subdivided according to their sampling strategy. The first class determines new textures by matching global statistics in feature space. The second class generates pixels based on local information in the sample and result texture.

A number of methods following a *global sampling* approach have been developed. One of the first – inspired by the work of Julesz – has been reported in [92]. A pixel-wise optimization strategy is followed, minimizing a goal function with respect to selected statistical features. Given a sample texture T_{src} and a target texture T_{tgt}, pixels in the latter are changed so as to minimize the error E between the respective feature vectors $\mathbf{p}_{src}$ and $\mathbf{p}_{tgt}$. One key element of the approach is the appropriate definition of the feature vectors. One option is to model the texture by pixel pair co-occurrences at certain distances, thus capturing second-order statistical properties. Unfortunately, in such a straightforward implementation, the computation time depends quadratically on the number of texture colors. As an alternative, it was suggested that the full co-occurrence matrices be replaced with simpler models based on moments of various order, for instance, second-order spatial moments combined with first-order spatial averages (i.e., autocovariance with intensity histograms). However, this resulted in reduced synthesis quality.

A different paradigm was followed in [117], inspired by psychophysical studies which assume that a bank of space localized filters are underlying human texture perception to discriminate texture fields. The method consists of matching histograms at different levels of a Laplacian or steerable texture pyramid. The first phase is the analysis of a sample texture, which consists of building an image pyramid where the levels are obtained by convolution and subsampling. The same is done for a white noise image, which is modified iteratively in the second phase to match the appearance of the sample texture. This is achieved by multiscale matching of marginal histograms. While the Laplacian pyramid is suited for isotropic textures, the steerable version enables the synthesis of textures with some oriented structures. Although there is no proof of convergence, reasonable results are obtained after a few iterations. Nevertheless, the technique performs only well on basic stochastic homogeneous textures. Moreover, performing too many matching iterations can introduce artifacts in the synthesized image.

A method related to the previous work was introduced in [31]. Again a multiresolution filter-based approach is applied, following a coarse-to-fine manner. The algorithm randomizes an input texture sample while preserving the cross-scale dependencies. First an analysis pyramid is constructed based on responses to a bank of oriented first- and second-order Gaussian derivatives. Then the synthesis is carried out on subsequent levels by building a set of candidate pixels for a specific texture location. The candidates are selected if the distances of features between the analysis pixel feature vector and the synthesized feature vector is below a set of selected thresholds. Difficulties in

this method are due to the nonintuitive tuning of the threshold parameters; especially, since the outcome is very sensitive to this choice. Additionally, boundary artifacts are produced when the textures are not tileable.

Another approach following a multiresolution paradigm has been suggested in [244, 217]. This method is based on matching first- and second-order properties of wavelet coefficients and coefficient pairs of steerable pyramids. Pixels in the image pyramid are updated according to iterative orthogonal projections from the filter response of the synthetic texture onto that of the sample.

A combination of multiscale feature matching with Markov Random Field (MRF) methods – the FRAME model – has been presented in [309, 308]. It builds on comparing marginal histograms of Gabor filter bank responses as features. The synthetic texture is updated according to MRF probability functions. While a solid mathematical framework is presented, the method is computationally quite expensive.

A difficulty, which hampers the application of all these multiscale approaches in our specific texture generation process is the masking of pixels. During the convolution steps between the different subbands, unknown texture values have to be propagated, which reduces the quality of the overall synthesis process.

For completeness, the work described in [100, 101] should also be mentioned. Instead of matching global features, the authors suggest performing a global spectral analysis of a texture sample in order to obtain basis and perturbation functions for procedural texture generation. A 2D texture sample T_s is approximated by using a summation of cosines.

$$r(x,y) = A_0 + \sum_{i=1}^{N} A_i \cos\left(2\pi(f_i x + g_i y) + \theta_i\right). \qquad (3.12)$$

Here, A_0 is the average value of the texture, A_i the amplitude of the i-th significant term, f_i, g_i its frequencies, and θ_i its phase, respectively. The cosines are determined by spatial partitioning of the spectrum of T_s and selecting the dominant signal in each bin. A new texture T_n, which can be 2D or 3D, can then be synthesized by using a pseudorandom signal. A problem is the control of the variation along the third dimension, since the texture remains a 2D pattern, which is perturbed along the third axis. This situation can, however, be partly alleviated by using additional samples representing different orthogonal slices of a structure.

In contrast to the methods discussed above, several approaches follow *local sampling strategies*. The majority of these algorithms model texture as a Markov Random Field and generate new images by stochastic sampling.

A core assumption of MRF is *locality*, that is, the value of a pixel is predictable from a finite set of neighboring pixels, and independent of the rest. This property of spatial Markovianity is described by the local conditional probability density function (LCPDF) (see, e.g., [24, 97]).

$$P(X_s = x_s | X_r = x_r, r \neq s) = P(x_s | x_r, r \in \mathcal{N}_s), \qquad s \in S, x_s \in \mathcal{L} \quad (3.13)$$

where $x_s \in \mathcal{L}$ is the state of the variable X_s at site s on texture lattice S, and $\mathcal{N}_s \subset S$ is a neighborhood of s. The second property of MRF is *stationarity*. This usually refers to the fact that the marginal probability of any texture window remains invariant under translation.

In order to calculate 3.13, the equation is reformulated according to the Hammersly-Clifford theorem [24]. Under positivity condition on a finite lattice, an MRF can be rewritten as a Gibbs random field, which assigns a probability to each realization $X = x$ in the MRF.

$$P(X = x) = \frac{1}{Z} e^{\left(-\sum_{C \subset S} V_C(x)\right)}. \qquad (3.14)$$

Here, V_C assigns a potential based on site values to the clique C, which is a subgraph of S where all sites are neighbors, and Z is a normalizing constant. The conditional probability can then be obtained using the defined potential function.

$$P(X_s = x_s | X_r = x_r, r \neq s) = \frac{e^{\left(-\sum_{s \in C} V_C(x)\right)}}{\sum_{\lambda \in \mathcal{L}} e^{\left(-\sum_{s \in C} V_C(\lambda, x_{\neq i})\right)}}. \qquad (3.15)$$

New texture can thus be synthesized by drawing samples from the distribution P. Two distinct strategies can be identified to carry out this step. *Parametric* approaches try to learn the parameters of the assumed underlying Gibbs distribution from samples. Unfortunately, this is hampered by the high dimensionality of the approach. The main problem is the fitting of all parameters for larger neighborhood sizes, which is computationally intractable. Therefore, *nonparametric* methods refrain from explicitly determining the MRF model.

The former approach was applied in [62]. The authors assumed autobinomial conditional probabilities for the clique potentials as a texture model. Second-order statistics of particular textures were compared to those predicted by the MRF.

A cluster-based, semiparametric model was introduced in [216]. The probability function was compressed with discretized Gaussian kernels.

$$P(x) = \sum_{i=1}^{N} w_i \left(\prod_{j=1}^{M} G_{i,j}(x) \right), \qquad w_i > 0, \sum w_i = 1 \qquad (3.16)$$

where w_i are weighting parameters. According to the restricted probability function, the value of a new pixel is then determined in a fixed order within a causal neighborhood. Moreover, to reduce complexity, a hierarchical approach is applied.

In contrast to the previously discussed methods, nonparametric texture synthesis approaches do not require the direct estimation of parameters in statistical models.

Initial work in this direction has been described in [204]. The authors developed a noncausal, nonparametric, multiscale MRF model capable of synthesizing a large range of textures. The local conditional probability density function was determined to obtain the probability of a pixel value in a noncausal neighborhood. A nonparametric estimate of the LCPDF of a sample image was obtained by building its multiscale histogram. A Parzen-window density estimator is applied to the multiscale histogram in order to determine the LCPDF. Pixelwise stochastic relaxation is then carried out to synthesize a new texture. Moreover, they augmented their coarse-to-fine approach with local annealing and parallelization of the relaxation.

In [79] a nonparametric sampling approach was described. Assuming an MRF model, a probability distribution for a target pixel is obtained based on its already synthesized neighbors, and on windows of the sample texture with similar neighborhoods. An exhaustive nearest neighborhood search has to be performed for each target pixel. A new image is grown in a spiral, one pixel at a time, starting from an arbitrarily placed 3x3 seed. In general, the neighborhood window has to be large enough to capture local structures completely. Moreover, problems can occur when no good matches can be found. The algorithm can get lost in the search space and start sampling randomly which produces incorrect results.

A similar technique was suggested in [290]. Again, no explicit probability distribution was constructed. The authors synthesized a new texture pixel by pixel in scanline order, while preserving local similarity. This was done by considering an L-shaped neighborhood of fixed size in the current output image, and searching the input sample for the candidate with the most similar neighborhood, which was then copied into the output image. Again, this method is quite slow, due to the extensive searching process. Therefore, the authors suggest accelerating the neighbor search with tree-structured vector quantization (TSVQ). A further improvement in computation time could be achieved by incorporating a multiresolution pyramid. These enhancements make the algorithm two orders of magnitude faster then the previous one. Unfortunately, the method tends to introduce some blurring into the results.

A further improvement of the nearest neighbor search is described in [8]. It is based on the observation that neighbors to a pixel in the output image often come from locations in the sample image which are near to the sources of pixels in its current neighborhood. Thus instead of an exhaustive search, a limited number of candidates is generated, based on the original locations of the current neighborhood pixels. To this end, a coherence map is maintained, which stores source locations of synthesized pixels.

Patch-Based Synthesis

In recent years a different paradigm for texture synthesis has emerged. No explicit mathematical texture model is developed, but instead a heuristic approach is followed, in which new textures are obtained by copying complete patches from example images. Usually, this is performed in an iterative fashion, where new patches are selected according to some error metric. The latter ensures an optimal fitting of overlap regions at the patch edges. The overall process can be influenced by selecting the patch size or the extent of the overlap regions. The main focus of most methods is the treatment of the patch boundaries, as well as employing optimal search strategies to select the patches.

The first work following a patch-based strategy – termed *chaos mosaics* – was reported in [300]. Instead of copying individual pixels, the authors randomly placed source texture patches in the output image. Their biggest problem was feature discontinuities across seams. In later work they overcame some of these problems with simple cross-edge filtering [299]. A key feature of their method was the fast synthesis of output images with reasonable quality.

Related to the previous method, in [219] *lapped textures* are introduced, which can be used to cover arbitrary triangle meshes with example textures. Seams across patches are avoided with alpha blending. A robust flattening approach is used to minimize distortions and match local orientation.

A synthesis process termed *image quilting* is outlined in [80]. New textures are synthesized one patch at a time in scanline order by placing overlapping square samples from a source image. Candidate patches are selected randomly from the source so as to minimize the L2 norm for pixels in the overlapping area. Once a square patch has been found, an optimal cutting patch between adjacent patches is determined. The authors apply dynamic programming to perform this minimum error boundary cut.

Similarly, in [161] another patch-based sampling strategy is suggested. Instead of determining an optimal cut between patches, boundary errors are alleviated by simple blending. In addition, a number of algorithmic improvements are made to enable real-time user interaction. An approximate nearest neighbor search scheme is followed for patch selection. Principal component analysis is applied to reduce the dimensionality in this step. Moreover, further enhancements could be achieved by using hierarchical data structures, e.g., kd-trees to find the nearest neighbor or quad-trees to select initial candidates.

Improved handling of patch overlap regions used in a hybrid synthesis approach is described in [192]. A multiscale paradigm is followed to minimize the border error. After a candidate patch is selected, the error of the overlap region is examined. If it exceeds a predefined threshold, a smaller patch is resynthesized and tested again. If necessary, this process is iterated down to individual pixels. In order to speed up the candidate search, it is carried out in the Fourier domain. This allows a quick calculation of the error with a fast Fourier transform (FFT).

In [150] the treatment of the patch boundaries is formulated as a graph cut problem. By representing the overlap region as a graph, the task of finding the best seam turns into the search for a minimum-cost graph cut. Again, an FFT-based acceleration is applied by carrying out patch searching in the Fourier domain.

Patch-based texture synthesis provides good results at reasonable processing times; however, a number of limitations exist which make their usage complicated in our application area. One problem is due to the image masks present in our texture samples, which limit the candidate search. Another drawback is the recalculation of texture patches necessary if affine transformations are to be applied to the texture, which reduces interactivity during the scene generation process. Finally, due to the patch oriented nature of these methods, stochastic variation can be diminished compared to pixel-wise strategies. For these reasons, pixel-wise synthesis approaches are mainly employed. In the following two sections, the methods applied to generate textures for laparoscopic as well as hysteroscopic scenes are discussed in more detail.

3.6.3 Texture Generation for Laparoscopic Simulation

In the context of simulation of surgical interventions in laparoscopy, a two-stage texture generation process can be followed [182]. The main step consists of pixel-based texture synthesis. It provides base textures of moderate stochastic variation, which is sufficient for the majority of the abdominal organs. If required, these textures can then be enhanced by adding low-frequency detail in a second step, following a procedural approach.

The synthesis method applied is the one detailed in [92]. Inspired by Julesz' conjecture, it models texture by pixel-pair statistics. An advantage of this approach is its capability of dealing with masking, which is often necessary due to the low quality of our source images. Moreover, the method can easily be extended to 3D synthesis, which simplifies the required texture mapping to object meshes. Furthermore, the technique allows the integration of directional control of the synthesis process, which permits the generation of anisotropic textures. Finally, varying levels of synthesis quality can be achieved by applying different statistical models for determining the texture feature vectors. This allows for the selection of an appropriate model depending on the character of the sample image, while the texture generation procedure stays the same.

The main idea of this approach is to minimize the error between feature vectors containing a statistical description of a source and a target texture. First, for a given sample image T_{src}, the feature vector $\mathbf{p}_{src}$ is determined. Similarly, the vector $\mathbf{p}_{tgt}$ of an arbitrary sized target texture T_{tgt} is obtained. Following an optimization in a least squares sense, pixels in the output image are changed, so as to minimize the error between the feature vectors.

$$E = \|\mathbf{p}_{tgt} - \mathbf{p}_{src}\|^2 \overset{!}{=} min \, . \tag{3.17}$$
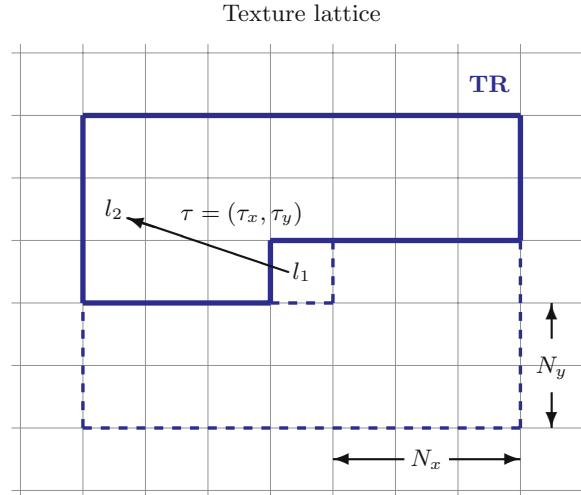
Texture lattice

Figure 3.11. Translation window TR for determining co-occurrences

To initialize this procedure, the target texture is filled with noise, which should match the histogram of that of the source image. This can be done by randomly copying pixels of the sample texture to the target lattice. While this is only an approximation of the source histogram, the resulting deviation does not influence the synthesis approach. Moreover, since first-order statistics are part of the applied texture models, this will automatically be adapted during the optimization. Finally, the random initialization ensures variation of the output samples.

After this initial setup step, the algorithm iteratively visits locations in the output lattice in a random order. The current texel value l is replaced with the value l', which maximally decreases the squared Euclidean feature vector difference. Recomputing the whole vector $\mathbf{p}_{tgt}$ to determine the error E would tremendously decrease performance. However, this is not necessary. Instead, it is sufficient to just determine the change of error Δ_E resulting from replacing l with l' in the target texture.

$$\Delta_E = E(l) - E(l') . \qquad (3.18)$$

Thus for each location the algorithm visits, we have to determine the value l', which maximizes Δ_E. If such a value can be found, l is replaced by l', and the feature vector $\mathbf{p}_{tgt}$ as well as the error E have to be updated.

Following the outline of the algorithm, we now have to define the statistical models which should be used to determine the feature vectors. While a number of approaches would be possible, this discussion focuses on the co-occurrence and the autocorrelation models.

Co-occurrence Model

This model is based on the co-occurrence of pixel color pairs in the texture lattice with a relative distance of τ. This translation vector is usually encoded with the tuple (τ_x, τ_y) denoting the L_1-distance in the x- and y-direction, respectively. The probability of co-occurrence of colors $l_1, l_2 \in \mathcal{L} = \{1, \ldots, L\}$ at distance τ is given by:

$$r_\tau(l_1, l_2) = \frac{1}{N} \sum_{i=1}^{N} \delta(t_i - l_1) \cdot \delta(t_{i+\tau} - l_2) \,. \tag{3.19}$$

Here, N is the total number of texture pixels t in the image lattice. They are indexed in 1D fashion, with t_i being at location (x, y) and $t_{i+\tau}$ at location $(x + \tau_x, y + \tau_y)$. By determining all probabilities for all colors $l_1, l_2 \in \mathcal{L}$ for a specific translation τ_k, we can setup a co-occurrence matrix.

$$\mathbf{M}_2(\tau_k) = \begin{pmatrix} r_{\tau_k}(l_1, l_1) & \cdots & r_{\tau_k}(l_1, l_{|\mathcal{L}|}) \\ \vdots & \ddots & \vdots \\ r_{\tau_k}(l_1, l_{|\mathcal{L}|}) & \cdots & r_{\tau_k}(l_{|\mathcal{L}|}, l_{|\mathcal{L}|}) \end{pmatrix} \,. \tag{3.20}$$

By combining all matrices $\mathbf{M}_2(\tau_k)$ for all translation vectors τ_k in the texture lattice T, we could build the feature vector $\mathbf{p}$. However, its size linearly depends on the number of translations and quadratically on the number of colors. Therefore, some reductions in size have to be made to keep the problem manageable. The translations can be restricted to a certain distance. This is done by considering just a limited window of translations. Moreover, by exploiting the fact that $r_\tau(l_1, l_2) = r_{-\tau}(l_2, l_1)$, we can further reduce the number of translations. Hence, the translation window (see Figure 3.11) is defined according to

$$TR = \left\{ \tau = (\tau_x, \tau_y) \middle| \begin{array}{l} (0 \leq |\tau_x| \leq N_x \wedge 1 \leq \tau_y \leq N_y) \quad \vee \\ (-N_x \leq \tau_x \leq -1 \wedge \tau_y = 0) \\ \tau_x, \tau_y, N_x, N_y \in I\!N \end{array} \right\} \,. \tag{3.21}$$

Therefore, the number of parameters in the vector needed to describe textures by using second-order spatial averages is finally given by $|\mathcal{L}|^2 \cdot (2N_x N_y + N_x + N_y)$. After setting up the initial feature vectors, the algorithm randomly selects positions t_i in the lattice of the target texture. For these positions the texture value which maximizes Δ_E has to be found. Since we limited the translations to the window TR, we only have to examine the positions $t_{i+\tau}$ and $t_{i-\tau}$ for a specific translation τ. Instead of recomputing the whole error, we can just update the error change. If l^+ is the color at $t_{i+\tau}$, and l^- the one at $t_{i-\tau}$, then a replacement of l with l' at t_i necessitates the replacement of

co-occurrence (l, l^+) with (l', l^+), and (l^-, l) with (l^-, l') in the feature vector $\mathbf{p}_{tgt}$. Thus, the vector update for the translation τ can be carried out following the scheme

$$r_\tau^{tgt}(l, l^+) := r_\tau^{tgt}(l, l^+) - \frac{1}{N_\tau^{tgt}} \tag{3.22}$$

$$r_\tau^{tgt}(l', l^+) := r_\tau^{tgt}(l', l^+) + \frac{1}{N_\tau^{tgt}}$$

$$r_\tau^{tgt}(l^-, l) := r_\tau^{tgt}(l^-, l) - \frac{1}{N_\tau^{tgt}}$$

$$r_\tau^{tgt}(l^-, l') := r_\tau^{tgt}(l^-, l') + \frac{1}{N_\tau^{tgt}} .$$

Here, N_τ^{tgt} denotes the total number of pixel-pairs lying inside the output image lattice for the specific translation τ. This update has to be applied for all $\tau \in TR$. Based on the vector updates of Equation 3.22, the resulting contribution to the error change can be determined. When replacing l with l' the error change is given by

$$\Delta_E(l') = \sum_{\tau \in TR} \left[\frac{4}{(N_\tau^{tgt})^2} + \frac{2}{N_\tau^{tgt}} \cdot \right.$$
$$\left(r_\tau^{src}(l, l^+) - r_\tau^{tgt}(l, l^+) + r_\tau^{tgt}(l', l^+) - r_\tau^{src}(l', l^+) + \tag{3.23} \right.$$
$$\left. \left. r_\tau^{src}(l^-, l) - r_\tau^{tgt}(l^-, l) + r_\tau^{tgt}(l^-, l') - r_\tau^{src}(l^-, l') \right) \right] .$$

Using this equation, we can now determine the value $l^* \in \mathcal{L} - \{l\}$, which maximizes Δ_E. If the error change is less than or equal to zero, then we already have found the optimal value and can continue with a new location in the lattice. Otherwise, we replace l with l^*, and update $\mathbf{p}_{tgt}$ according to Equation 3.22, as well as the error $E := E + \Delta_E(l^*)$. It should be noted that in order to deal with the boundary, the texture is regarded topologically as a torus by joining opposite edges. This also produces output images, which tile seamlessly. Example textures synthesized with this algorithm can be seen in Figure 3.12.

While the convergence of the algorithm has not been proven theoretically, in practice the error monotonically decreases with each scan. The lower bound of the error is zero; however, one can not guarantee a perfect fit of the co-occurrences (i.e., zero error). Typically, the optimization can be stopped after 6 iterations, where only a small percentage of pixels is changed. This change rate can actually be used as a termination criterion. Moreover, it should be noted that a fundamental element of the method is the random selection of pixel locations. If a scanline order visitation scheme were followed, the error decrease would not be homogeneous over the whole domain, which would become visible as artifacts in the output texture.
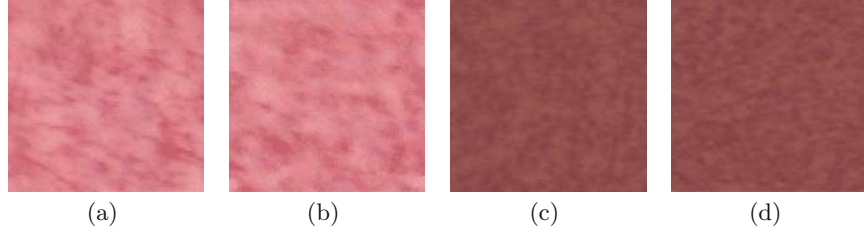
<center>(a)   (b)   (c)   (d)</center>

Figure 3.12. Texture synthesis with co-occurrence model. (a) Drift-corrected, quantized uterus texture sample, (b) Synthesized uterus texture, (c) Drift-corrected, quantized liver texture sample, (d) Synthesized liver texture

A major drawback of the texture model is the large number of parameters in the optimization. In order to synthesize textures of sufficient quality, the translation window has to be large enough. This leads to a considerable increase in computation time, which renders the method unfit for interactive scene definition purposes. A reasonable alternative is the autocorrelation model, which replaces the full co-occurrence matrices with second-order spatial moments.

Autocorrelation Model

According to the Wiener-Khintchine theorem, the Fourier transformation of the autocorrelation function describes the power spectrum of a signal.

$$J(\omega) = \int_{-\infty}^{\infty} K(\tau)e^{-i\omega\tau}d\tau \tag{3.24}$$

where the autocorrelation is defined as

$$K(\tau) = \frac{1}{N\sigma^2}\sum_{i=1}^{N}(t_i - \mu)(t_{i+\tau} - \mu) \tag{3.25}$$

with σ^2 denoting variance and μ mean of the texture signal. The theorem indicates that the autocorrelation function contains some information about frequency and orientation in a texture. Therefore, it seems appropriate to use it as a reduced statistical model for describing texture. However, since it only provides the statistical moments, the histogram of the texture also has to be taken into account by determining

$$H(l) = \frac{1}{M}\sum_{i=1}^{M}\delta(t_i - l) \qquad \forall l \in \mathcal{L} \tag{3.26}$$

with M being the number of pixels in the texture lattice. Combining the histogram and the autocorrelation values for translations $\tau \in TR$, we form

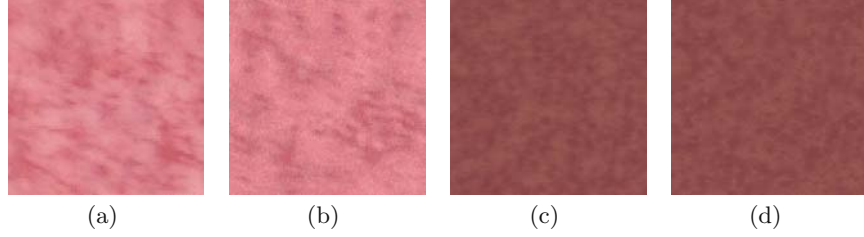(a)   (b)   (c)   (d)

Figure 3.13. Texture synthesis with autocorrelation model. (a) Drift-corrected, quantized uterus texture sample, (b) Synthesized uterus texture, (c) Drift-corrected, quantized liver texture sample, (d) Synthesized liver texture

feature vectors $\mathbf{p}_{src}$ and $\mathbf{p}_{tgt}$. For this model, the length of the vectors is reduced to $|\mathcal{L}| + (2N_x N_y + N_x + N_y)$.

The different definition of the feature vector also necessitates a new error metric. Optimally, the histograms H_{src} and H_{tgt} should be identical; however, due to the size of the optimization space, constraint optimization is not practicable. Therefore, we minimize the error according to:

$$E = \|\mathbf{p}_{src}^K - \mathbf{p}_{tgt}^K\|^2 + \alpha\|\mathbf{p}_{src}^H - \mathbf{p}_{tgt}^H\| \overset{!}{=} min \qquad (3.27)$$

where $\mathbf{p}_{<>}^K$ and $\mathbf{p}_{<>}^H$ denote the subvectors containing the autocorrelation and histogram values, respectively. The parameter α has to be selected manually to control histogram equality. As in the previous model, optimization is carried out by maximizing the error change. When replacing pixel l with l' at position t_i, first the histogram has to be updated:

$$H_{tgt}(l) := H_{tgt}(l) - \frac{1}{M_{tgt}} \qquad (3.28)$$

$$H_{tgt}(l') := H_{tgt}(l') + \frac{1}{M_{tgt}} \;.$$

Thereafter, the autocorrelation has to be recomputed. Again, for a specific translation vector τ, values l^- and l^+ are affected.

$$K_{tgt}(\tau) := K_{tgt}(\tau) + \frac{1}{N_\tau \sigma^2}(l' - l)(l^- + l^+ - 2\mu) \qquad (3.29)$$

where σ is the standard deviation, and μ the mean of the texture lattice T. As in the previous model, we also have to determine the resulting contribution to the error change, when replacing l with l'. Since in general α should be set high enough to ensure histogram equality, we can assume that only minute changes of the mean value μ will happen. By considering the value as being constant, we can avoid the complete recomputation of the moments, and again perform local error updates.
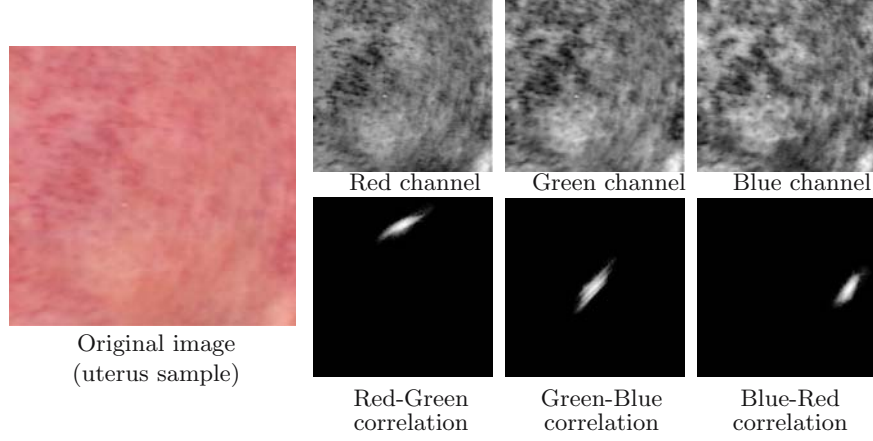
Red channel   Green channel   Blue channel

Red-Green   Green-Blue   Blue-Red
correlation   correlation   correlation

Original image
(uterus sample)

Figure 3.14. Correlation of color channels

$$
\begin{aligned}
\Delta_E(l') = \frac{2\alpha}{M} &\left(H_{src}(l) - H_{tgt}(l) - H_{src}(l') + H_{tgt}(l') + \frac{1}{M} \right) \\
+\ &\frac{l'-l}{\sigma^2} \sum_{\tau \in TR} \left[\frac{1}{N_\tau^2 \sigma^2}(l'-l)(l^- + l^+ - 2\mu)^2 \right. \\
-\ &\left. \frac{2}{N_\tau}(l^- + l^+ - 2\mu)(K_{src}(\tau) - K_{tgt}(\tau)) \right].
\end{aligned}
\tag{3.30}
$$

After determining the optimal value $l^* \in \mathcal{L} - \{l\}$, the same update strategy is followed as before. In general, the quality of the synthesis process is reduced due to the model simplification. However, reasonable results can still be achieved due to the rather stochastic nature of the organ textures. Examples synthesized using the same sample images as with the previous method are depicted in Figure 3.13.

Algorithm Enhancements

Several improvements or extensions are possible based on the texture synthesis approaches discussed above. Some of these will be outlined in this section.

One important aspect is the *treatment of color* in the synthesis methods. So far, the discussion has been limited to samples using small numbers of indexed colors resulting from a quantization step. While reasonable results can be achieved with these reduced color models, the usage of full RGB images as source textures for the synthesis process should also be examined. The straightforward approach would be to reformulate the synthesis algorithm for three-dimensional vectors containing the red, green, and blue channel of the image. Unfortunately, due to the high correlation between the channels, a
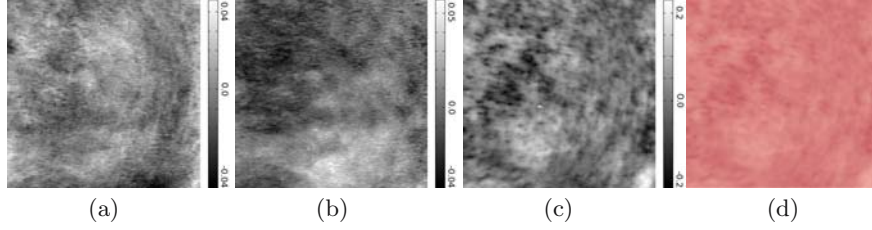
(a)   (b)   (c)   (d)

Figure 3.15. Decorrelation of color channels (normalized view). (a) New color channel 1, (b) New color channel 2, (c) New color channel 3 (primary channel), (d) Color image based on primary channel

separate treatment of individual channels is not possible. An example of the color correlation for a typical texture can be seen in Figure 3.14. The top row shows the red, green, and blue components of the source image, while the bottom row depicts the color distribution projected on the Red-Green, Green-Blue, or Blue-Red surface of the RGB cube, respectively. A clear linear correlation between the color channels is visible.

One option to amend this situation is the linear decorrelation of the color channels via the Principal Component Analysis, thus defining a new, texture-specific color space. This can be done by determining the eigenvalue decomposition

$$\mathbf{C} = \mathbf{P}\tilde{\mathbf{C}}\mathbf{P}^T \tag{3.31}$$

of the texture covariance matrix

$$\mathbf{C} = \frac{1}{N}\sum_i (\mathbf{T}_i - \boldsymbol{\mu})(\mathbf{T}_i - \boldsymbol{\mu}^T) \tag{3.32}$$

where $\mathbf{T}_i$ is the RGB value of texel i, $\boldsymbol{\mu}$ the vector of channel means of texture $\mathbf{T}$ containing N texels. Matrix $\mathbf{C}$ is square, real, and symmetric, with orthogonal eigenvectors. Matrix $\mathbf{P}$ defines a linear transformation from color space $\mathcal{C}$ into color space $\tilde{\mathcal{C}}$. Since $\tilde{\mathbf{C}}$ is diagonal, the three color channels are linearly decorrelated. By subtracting the mean $\tilde{\boldsymbol{\mu}}$ from the decorrelated color channels, we obtain a new color space with mean values zero. However, it should be noted that the PCA captures only linear correlation to reduce dimension, but fails to detect nonlinear components. Since some nonlinear dependency is also present in organ textures, the new color channels are not fully independent. Nevertheless, the nonlinear component is in general negligible. Figure 3.15 depicts the obtained linearly decorrelated color channels for the previously used example texture. Moreover, for the highly correlated organ textures, it can usually be noticed that a primary channel exists, which contains most of the texture color information. This allows one to focus the synthesis process only on this channel. The final texture can then be obtained by setting the other
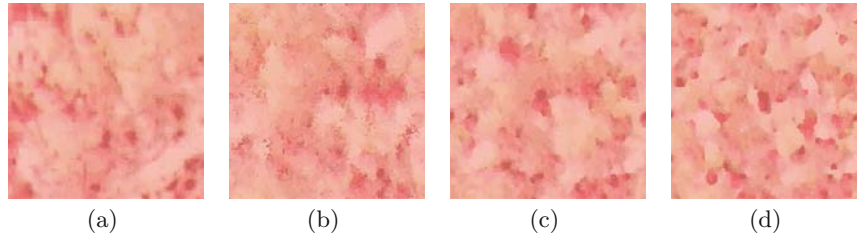
(a)   (b)   (c)   (d)

Figure 3.16. Synthesis examples using distance weighting with varying β. (a) Original texture of endometrium, (b) Synthesized with $\beta = 0.0$ (unmodified version), (c) Synthesized with $\beta = 2.0$, (d) Synthesized with $\beta = 3.0$

two channels to zero, and transforming the primary channel back into color space $\mathcal{C}$. An example of the backtransformation of the primary channel is also visualized in Figure 3.15. The resulting texture can not easily be distinguished from the original one.

An additional option to control the texture synthesis process is the integration of a *distance weighting* function. A problem of the general formulation is the unbalanced contribution of different translations τ_k to the optimization. Since a larger number of texels is included in the computation for longer translations, mainly coarse structures are formed in the initial optimization steps. This results in inaccuracies of fine detail, which becomes apparent as slightly noticeable high-frequency noise in the image. To avoid this problem, in Equation 3.23 the contribution to the error update of different translations can be scaled with a weighting function $w(\tau_k)$. One possibility for the weighting could be a Gaussian filter. A simpler function, also providing good results, applies weighting according to

$$w(\tau_k) = \frac{1}{(\|\tau_k\|_2)^\beta} \; . \tag{3.33}$$

The synthesis process can be influenced by selecting parameter β. For the examined organ textures, values of $\beta \approx 2$ lead to improved synthesis output. An example of the influence of the parameter on the process is shown in Figure 3.16.

While the introduction of a weighting function is a rather heuristic step, a more formal approach can be taken using information theory principles to compute *transinformation*. This measure (sometimes also referred to as mutual information) denotes how much information can be obtained about one random variable by observing another. Generally, the transinformation of X relative to Y is given by

$$I(X, Y) = \sum_{x,y} p(x, y) \log \left(\frac{p(x|y)}{p(x)} \right) . \tag{3.34}$$

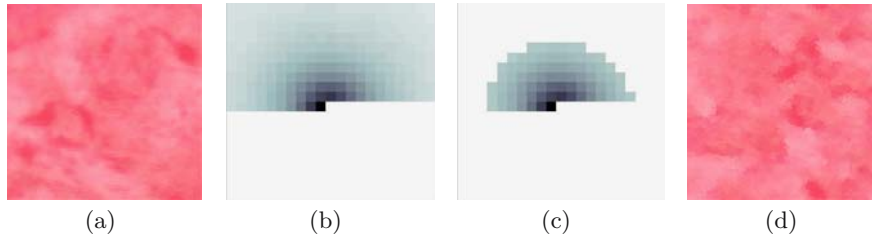(a)   (b)   (c)   (d)

Figure 3.17. Transinformation-based reduction of translation window. (a) Original uterus texture, (b) Transinformation for window TR with $N_x = N_y = 10$, (c) Reduced set of translations TR', (d) Synthesized texture with TR'

For the synthesis process, this model can be applied to determine how much information about a texture is contributed by a specific translation τ. Using the previously defined co-occurrence $r_\tau(l_1, l_2)$, as well as $r_\tau(l_1|l_2)$, which denotes the conditional probability of texel i having color l_1, given that texel $i + \tau$ has color l_2, we can compute the transinformation $I(i, i + \tau)$ for a specific τ by examining all pairs of colors $l_1, l_2 \in \mathcal{L}$. Determining this for all translations $\tau \in TR$, we can select only those which contribute more information than a user-defined threshold. This approach limits the number of translations, and thus also the overall computation time, while the synthesis quality is more or less maintained. An example is depicted in Figure 3.17. By removing all translations which contribute less than 20% of the maximal transinformation, the number of translations in a window $TR(N_x = N_y = 10)$ can be reduced from 220 to 77. It is interesting to note that the profile of the transinformation for the type of texture we examine is similar to the weighting applied in the previous method. Thus, due to the radially decreasing weights, the previous method also increases the influence of nearby texels, while more distant locations are neglected. By building a thresholded transinformation window, however, the number of computations can be significantly reduced. Unfortunately, for both methods some iterative tuning of the controlling parameters – weighting function or transinformation threshold – is required to obtain the desired results.

The synthesis techniques discussed so far have focused only on obtaining isotropic output images. However, for some specific organ surface patterns it would be useful to also allow generation of *anisotropic textures*. This is for instance true for the uterus, where texture is oriented along muscle fibers. In order to include anisotropy into the synthesis process, we have to extend the approach by considering the local coordinate systems in the texture, which define the orientation at that position. When considering a specific translation $\tau = (\tau_x, \tau_y)$ in window TR for the current texture location, we can apply a mapping to the local coordinate system via a rotation θ and optionally also a scaling (u_x, u_y). This gives a local set of translations $\tilde{\tau}$.

Rotated and scaled translation window at current location   Original texture   Synthesized aniso- tropic texture
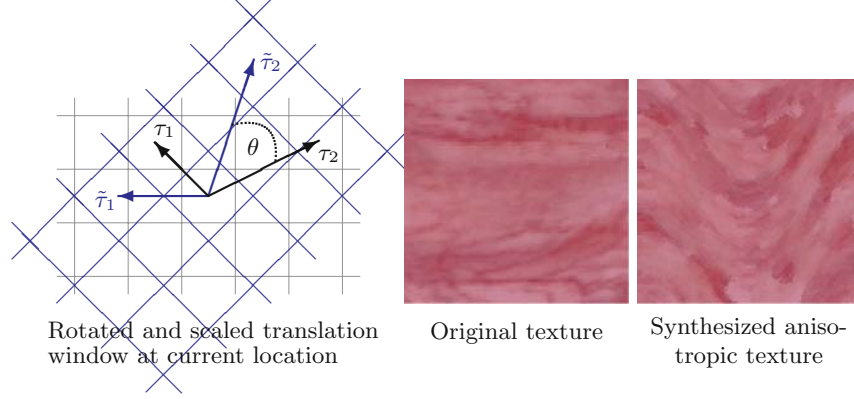
Figure 3.18. Rotation of translation window to include local orientation in the synthesis process. The example shows a sinusoidal wave overlaid on the texture lattice

$$\tilde{\tau}_x = u_x \tau_x \cos(\theta) - u_y \tau_y \sin(\theta) \qquad (3.35)$$
$$\tilde{\tau}_y = u_x \tau_x \sin(\theta) + u_y \tau_y \cos(\theta) \,.$$

Since the new translations usually do not coincide with the texture lattice, the color has to be bilinearly interpolated. If a quantization has been performed for the sample texture, then the new color might not be part of the histogram, and thus it has to be mapped to the closest color value. It should also be noted that for indexed color images, a bilinear interpolation cannot be done. Instead the color of the nearest neighbor has to be selected. An example of including local orientation into the synthesis is visualized in Figure 3.18. A sinusoidal wave is used to obtain local orientation θ at each lattice point, while no scaling is applied. A texture with prominent horizontal patterns is used as the source image to better visualize the effect. It has to be mentioned that for more isotropic textures, the visual effect is reduced and sometimes almost not visible.

While this process allows the inclusion of local orientation into the synthesis process, the question of how to obtain these local patterns still has to be addressed. The anisotropies have to match the actual geometry of the object to be textured. Thus an orientation field has to be defined on an arbitrary triangular mesh. In practice, a user would specify a few major orientations on the mesh, while the remaining ones are obtained from an optimization process, ensuring smooth transitions. An orientation at a specific vertex i can be described by a local coordinate system $(\mathbf{u}_i, \mathbf{v}_i, \mathbf{n}_i)$. Since $\mathbf{n}_i$ is the normal at the vertex, the remaining degree of freedom is rotation θ_i around it. Thus in an optimization process, the difference between the rotations of the local coordinate systems at vertices i and j, which are connected by edge (i, j), has to be minimized. A global energy term can be defined according to

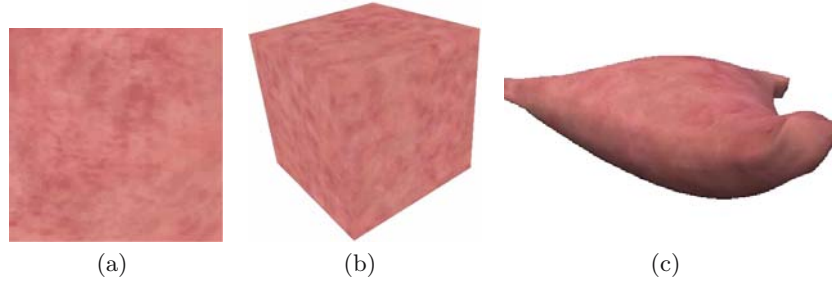(a)   (b)   (c)

Figure 3.19. Synthesis of isotropic 3D texture, (a) Original uterus texture, (b) Synthesized texture block, (c) Uterus surfaces carved out of 3D texture

$$E_\theta = \sum_{(i,j)} \frac{1}{|(i,j)|} \cdot \frac{1 - \cos\big(h(\theta_i, \theta_j)\big)}{2} \tag{3.36}$$

where $h(\theta_i, \theta_j)$ is a scalar valued function measuring the difference between the orientation of the local coordinate systems connected by an edge. After optimization by minimizing the energy term, a vector field with smooth transitions is obtained.

In the previous sections mainly 2D textures were considered; however, this approach can also be extended to generate *3D textures*. Assuming that the examined texture is isotropic, a statistical description can be built without taking the lattice orientation into account. Instead of using specific translation vectors τ, only the length of the vectors affects the model, thus making it independent of the dimension. Therefore, the statistical model for a specific 3D translation vector can be obtained from a 2D vector with the same length. It should be noted that vectors in space might occur for which a specific translation in 2D with corresponding length cannot be found. In these cases, the nearest neighbor can again be taken into consideration. With this simple extension, it is possible to synthesize isotropic 3D texture volumes. As mentioned, volumetric textures have the advantage that a mapping step to an object surface is not necessary. Figure 3.19 depicts a synthesized volume texture, which is mapped to the geometry of a uterine cavity. Nevertheless, computation times become quite high, even for smaller textures, thus making an interactive processing infeasible. Moreover, to obtain sufficient surface detail, high volume resolution is required. Due to texture memory limitations this cannot be achieved if several different textures are needed.

The described methods allow synthesis of organ textures of high quality. Nevertheless, a number of limitations were encountered. Using only highly homogeneous source images results in output images acceptable for highly realistic simulation. However, several organs show low frequency patterns on their surface, which cannot easily be replicated by the algorithm. Figure 3.20
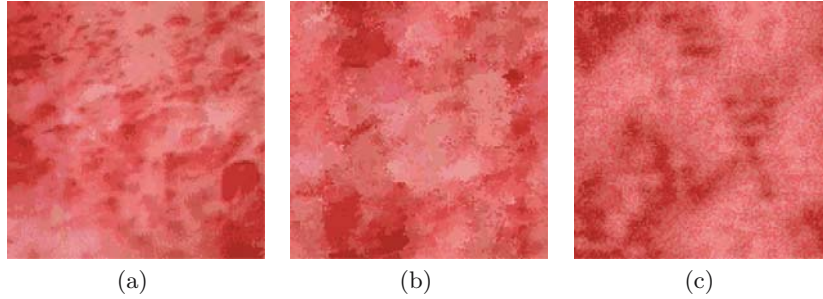
| | | |
| (a) | (b) | (c) |

Figure 3.20. Limitations of the synthesis models. (a) Drift-corrected, quantized uterus endometrium sample, (b) Synthesized with co-occurrence model, (c) Synthesized with autocorrelation model

depicts an example with synthesis outcome of insufficient quality for a sample with unhomogeneous patterns. Moreover, even with the optimized approaches, the computation time is still unfavorable. Finally, some iterative, and unfortunately also nonintuitive, parameter tuning is necessary to improve the outcome of the synthesis. With medical experts as the end users of the scene generation tool, this is not acceptable.

3.6.4 Texture Generation for Hysteroscopic Simulation

In order to be able to synthesize a wider range of textures, including inhomogeneities and anisotropies at low and high frequencies, a different approach has been integrated into the texture generation framework based on Markov random fields. A fast, multiscale nonparametric synthesis approach is followed, as outlined in [204], which captures sufficient higher-order statistical characteristics.

General MRF Model

In order to define the MRF, the pixels of an image $\mathbf{x}$ with dimension $M_x \times M_x$ are considered to be at sites s on a lattice $S = \{(i, j) \mid 0 \leq i, j < M_x\}$. Each site represents a variable X_s which is equal to a value x_s within the state space Λ. For grey value or color indexed images, the state space is defined as $\Lambda = \{0, 1, 2, \ldots, L - 1\}$, where L is the number of grey levels or colors. The configuration space for the sets of variables $\mathbf{X} = \{X_s \mid s \in S\}$ is the set of all possible images $\Omega = \Lambda^{M_x \times M_x}$. For a texture to be modeled as an MRF, it is assumed that the value of the pixels depends only on a limited number of local neighbors. In [24] it was proven that the joint probability measure on Ω is uniquely determined by its LCPDF with respect to a specific neighborhood system

$$P(x_s \,|\, x_r, r \neq s) = P(x_s \,|\, x_r, r \in \mathcal{N}_s) \tag{3.37}$$

where $\mathcal{N}_s$ represents a predefined neighborhood of site s. For homogeneous MRF a symmetric neighborhood of order c for a site s can be defined as

$$\mathcal{N}_s^c = \{(k,l) \in S \,|\, 0 < (k-i)^2 + (l-j)^2 \leq c\} \,. \tag{3.38}$$

For a parametric approach the LCPDF has to be estimated from the multidimensional histogram of a particular texture using Gibbs distributions. This allows us to define the probability density via appropriate parametric potential functions. The parameters have to be optimized to match the LCPDF to the histogram. Unfortunately, the sample data are sparsely dispersed over the corresponding multidimensional histogram and the true distribution is not known, which leads to results of lower accuracy when using parametric estimation. Therefore, a nonparametric approach is followed.

Nonparametric MRF Model

Given a texture sample $\mathbf{y} \in \Omega$ and a neighborhood system $\mathcal{N} = \{N_s \,|\, s \in S_y\}$, the first step of the nonparametric MRF is to determine the multidimensional histogram of the image. To this end, the frequency of occurance of a specific combination of pixel values L for all sites p using neighborhood $\mathcal{N}_p$ is obtained.

$$F(L_0, \ldots, L_{|\mathcal{N}_p|}) = \sum_{p \in S_y, \mathcal{N}_p \subset S_y} \delta(y_p - L_0) \prod_{r \in \mathcal{N}_p} \delta(y_r - L_{n_r}) \tag{3.39}$$

where n_r are indices for the sites r in neighborhood $\mathcal{N}_p$. By determining F for all combinations of $L_0, \ldots, L_{|\mathcal{N}_p|} \in \Lambda$, the multidimensional histogram can be built. The total number of dimensions is the statistical order of the model, which is equal to the neighborhood size $N = |\mathcal{N}_p|$. The LCPDF can now be calculated according to

$$P(y_s \,|\, y_r, r \in \mathcal{N}_s) = \frac{F(y_s, y_r, r \in \mathcal{N}_s)}{\sum_{\tilde{L} \in \Lambda} F(\tilde{L}, x_r, r \in \mathcal{N}_s)} \,. \tag{3.40}$$

It should be noted that the multidimensional histogram is usually only sparsely filled. If a 3×3 neighborhood were used with 32 grey levels, then the histogram would contain $32^9 \approx 3.51 \times 10^{13}$ bins. Even a large sample image with a resolution of 1024×1024 pixels would only fill a minute fraction of the histogram space. This situation can be alleviated by smoothing the distribution over the histogram. To this end a nonparametric density estimator can be applied. The Parzen-window estimator spreads each sample datum over a larger area in the histogram. Denoting $\mathbf{Z}_p = [y_p, y_q, q \in \mathcal{N}_p]$ and $\mathbf{z} = [x_s, x_r, r \in \mathcal{N}_s]$, the true density function can be approximated with
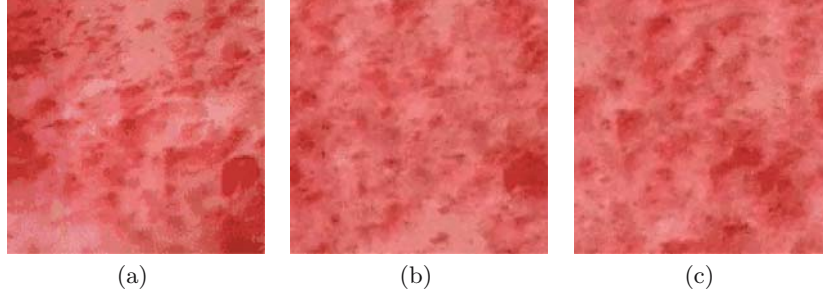
(a)   (b)   (c)

Figure 3.21. Synthesis examples with the nonparametric MRF model (3x3 neighborhood). (a) Drift-corrected, quantized uterus endometrium sample, (b) Synthesized texture example 1, (c) Synthesized texture example 2

$$\hat{F} = \frac{1}{nh^d} \sum_{p \in S_y, N_p \subset S_y} K\left\{\frac{1}{h}(\mathbf{z} - \mathbf{Z}_p)\right\} \tag{3.41}$$

where n is the number of sites $p \in S_y$, and $d = |\mathcal{N}_p| + 1$ the dimension of the histogram (i.e., the number of elements in $\mathbf{Z}_p$). The histogram is convolved with a radially symmetric Gaussian kernel.

$$K(\mathbf{z}) = \frac{1}{(4\pi^{d/2})} e^{\left(-\frac{1}{2}\mathbf{z}^T\mathbf{z}\right)} \tag{3.42}$$

The area of influence of the density estimator is controlled by the window parameter h. If it is too small, then noise will be introduced into the estimation function $\hat{f}$. If it is too large, then local detail will be blurred. According to [243], an optimal selection of the parameter can be done according to

$$h_{opt} = \sigma\left\{\frac{4}{n(2d+1)}\right\}^{1/(d+4)} \tag{3.43}$$

where σ^2 is the variance of the histogram of training image $\mathbf{y}$. The LCPDF can thus be approximated using the Parzen-window density estimator.

$$\hat{P}(x_s|x_r, r \in \mathcal{N}_s) = \frac{\sum_{p \in S_y, \mathcal{N}_p \subset S_y} e^{\left((-1/2h^2)(\mathbf{z}-\mathbf{Z}_p)^T(\mathbf{z}-\mathbf{Z}_p)\right)}}{\sum_{x_s \in \Lambda} \sum_{p \in S_y, \mathcal{N}_p \subset S_y} e^{\left((-1/2h^2)(\mathbf{z}-\mathbf{Z}_p)^T(\mathbf{z}-\mathbf{Z}_p)\right)}} \tag{3.44}$$

In order to synthesize a texture, stochastic relaxation can be applied using the estimation of the LCPDF. A popular approach for the relaxation is the *Iterative Conditional Modes* (ICM) method described in [25]. Starting with a randomly generated texture, the image sites are randomly visited and updated maximizing the marginal posterior distribution at each pixel. For small neighborhood sizes, superior results can be obtained using the nonparametric MRF model. In Figure 3.21, results from applying the method to the
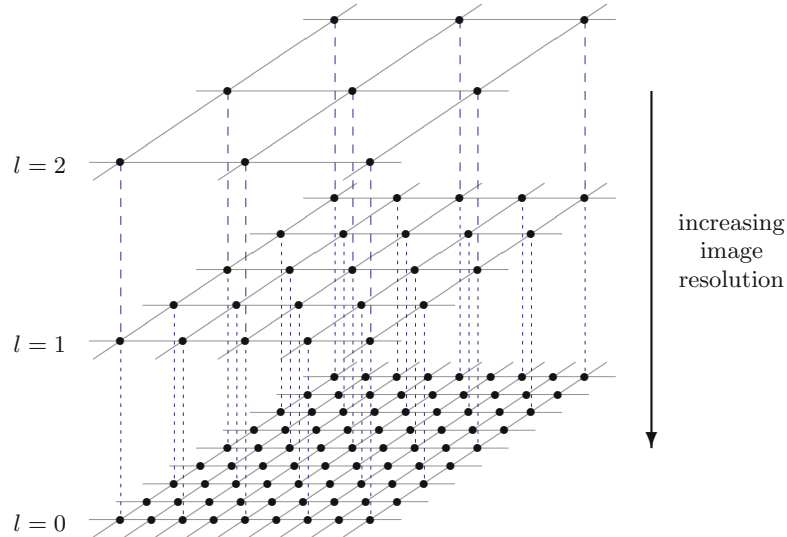
Figure 3.22. Decimation between grid levels in multiscale approach

previously shown unhomogeneous texture are depicted. One problem already encountered in the previous approach is the propagation of low-frequency, global image characteristics. These are typically propagated across the image lattice only by local interactions, and the relaxation can get stuck in local minima. To tackle this difficulty, a multiresolution technique can be followed.

Multiscale Texture Synthesis

The underlying idea is to synthesize coarse structures at lower resolution, and successively add more detail as the resolution increases. In this scheme, the outcome of the relaxation at one level constrains the relaxation at the following one. In addition to giving higher synthesis quality, this also reduces the number of iterations necessary to reach equilibrium. The multiscale representation is defined by images $\mathbf{X}^l$ for each level l on lattices given by:

$$S^l = \{(2^l i, 2^l j) \,|\, 0 \le i, j < N/2^l\} \tag{3.45}$$

where S^0 is the highest resolution image. The lattice S^{l+1} is obtained from S^l by decimation. The multigrid representation is depicted in Figure 3.22. Moreover, the neighborhood relationship for each level also has to be redefined:

$$\mathcal{N}_s^c(l) = \left\{(2^l m, 2^l n) \in S^l \,|\, 0 < (i - m)^2 + (j - n)^2 \le c\right\}. \tag{3.46}$$

Starting from the lowest resolution, at each grid level stochastic relaxation is carried out until equilibrium is reached. This requires that decimations $\mathbf{Y}^l$
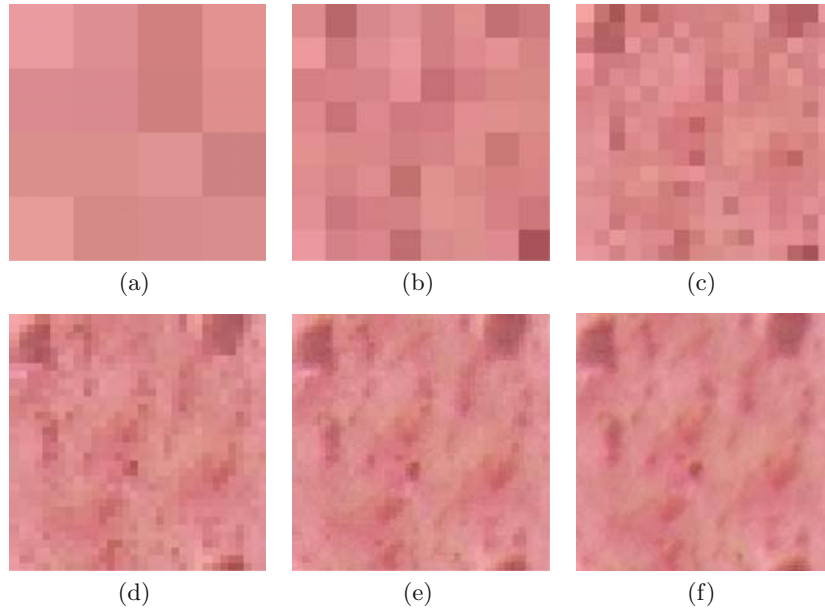
(a)   (b)   (c)

(d)   (e)   (f)

Figure 3.23. Multiscale texture synthesis. (a) Level 5 (4×4), (b) Level 4 (8×8), (c) Level 3 (16×16), (d) Level 2 (32×32), (e) Level 1 (64×64), (f) Level 0 (128×128)
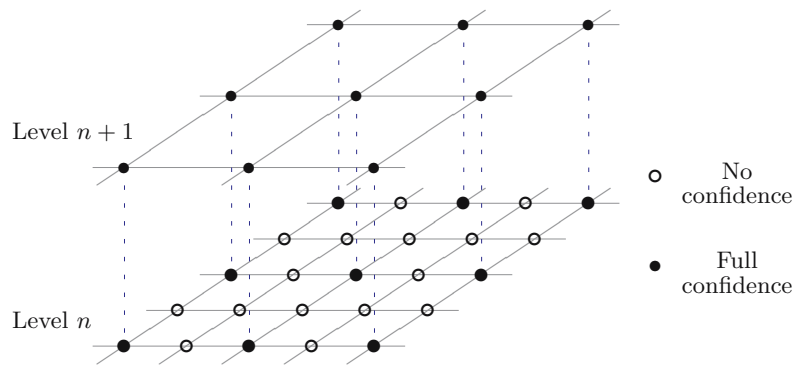

Figure 3.24. Propagation of pixel temperature between levels

of the training image are also obtained for all levels to construct the LCPDF. The individual equilibria of the multiscale approach during a synthesis process are shown in Figure 3.23.

Algorithm Enhancements

An extension of the synthesis approach can be made by including a *pixel temperature function* into the process. As with stochastic annealing, a local temperature function is integrated into the relaxation process. Each pixel is assigned a temperature with $0 \leq t_s \leq 1$, which represents a degree of confidence in the current value, where 0 denotes no, and 1 full confidence. The temperature is integrated into the Parzen-window estimation of Equation 3.41 by weighting the difference between the sample and output neighborhood values:

$$(\mathbf{z} - \mathbf{Z}_p) = \left[x_s - y_p, (x_r - y_r)(1 - t_r), r \in \mathcal{N}_s \right]. \tag{3.47}$$

After the relaxation process at level $l + 1$, all pixels propagated to level l receive complete confidence, while the rest have no confidence assigned (see Figure 3.24). Every time a pixel value is changed in the following relaxation on the new level, its temperature is adjusted according to

$$t_s = max\left\{ 0, \frac{1 + \sum_{r \in \mathcal{N}_s} t_r}{|\mathcal{N}_s|} \right\}. \tag{3.48}$$

At the start of the relaxation, only the values propagated from the previous level are used in the LCPDF. However, as the process advances, other sites gain more confidence, and affect the computation. Additionally, the confidence level can be used as a relaxation termination criteria, i.e., when all sites have reached full confidence, the relaxation can move on to the next level.

A further modification of the algorithm can be done by using a specialized *neighborhood searching scheme*. As previously discussed, an estimate of the LCPDF is obtained based on a sum of Gaussian kernels. Since the data are sparsely distributed, it can be assumed that the site giving the smallest distance between vectors $\mathbf{Z}_p$ and $\mathbf{z}$ will have the largest influence. Thus, the LCPDF can be approximated by just searching for the site with minimal distance. The sample subset is selected as all pixels with a neighbor of the same color as the respective neighbor of the output pixel being iterated. This is similar to the method proposed in [8]; however the subset is larger, resulting in a higher synthesis quality. Examples of the complete synthesis process can be seen in Figure 3.25.

3.7 Additional Texture Detail

With the nonparametric MRF synthesis strategy it is possible to synthesize textures showing relatively irregular patterns. However, some organ surfaces exist which need a specialized treatment, since not all their texture characteristics can be easily replicated. This includes, for instance, follicle patterns
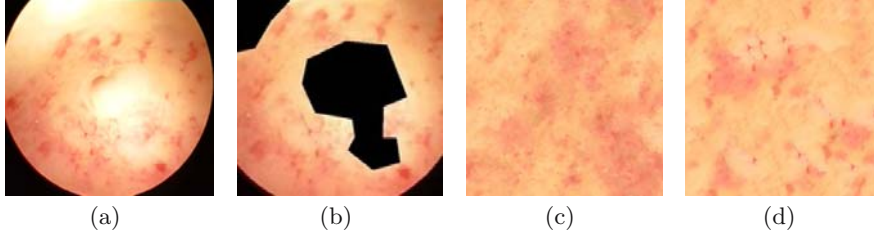
| (a) | (b) | (c) | (d) |

Figure 3.25. Fast MRF-based synthesis examples of in vivo texture (from [203]). (a) In vivo image, (b) Cropped and masked sample, (c) Synthesized texture 1, (d) Synthesized texture 2

on ovaries or vessel trees on some organ surfaces. Thus, an approach is required to further augment generated textures with macroscopic structural patterns. This can be done by procedural postprocessing of synthesized textures. The main idea is to optimally combine separate texture patterns, which necessitates the definition of texture transfer functions. As two examples, the enhancement of liver textures with low-frequency specks and ovary textures with follicle patterns will be discussed.

3.7.1 Embedding of Specks into Liver Textures

After generating a homogeneous base texture for the liver, it can be further enhanced by adding characteristic specks. Since these specks are usually too small to use a texture synthesis strategy, their appearance is generated using the basic procedural approach. To make the pattern more granular, the approach used for generating fat texture can be modified by steepening the edges of the noise signal with a transfer function defined as

$$\hat{T}_i = \frac{1}{2}\left(1 + \frac{\arctan\left((2s+1)^4(2T_i - 1)\right)}{\arctan(2s+1)^4}\right) \tag{3.49}$$

where T_i are the unmodified texture values, and s is a scaling factor ($0 \leq s \leq 1$). In order to merge the textures, a third, *mask texture* has to be created, which defines the smooth transfer between the base and pattern texture. For the specks, this mask texture can be automatically generated. To this end, spheres with varying radii are randomly distributed in the texture space. Moreover, to approximate the actual shapes of the specks, jitter, controlled by a vectorial noise function, is added. Finally, the textures are blended according to the mask texture. An example of this process is visualized in Figure 3.26. It should be noted that adding smaller detail like the specks is necessary only when dealing with relatively homogeneous base textures. More sophisticated algorithms, like the described MRF approach, are usually capable of automatically synthesizing these patterns.
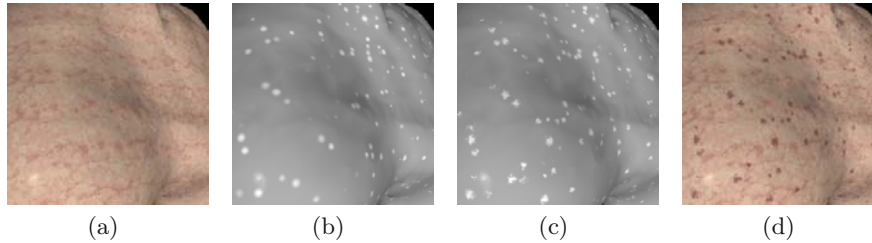
<div align="center">(a)　　　　　　　(b)　　　　　　　(c)　　　　　　　(d)</div>

Figure 3.26. Texture enhancement using blending. (a) Original textured surface, (b) Automatically generated, randomized spheres, (c) Jittered spheres according to noise function, (d) Blended 3D textures

3.7.2 Overlay of Real Follicle Textures

In some cases the embedding of additional texture is more complicated. This is for instance true if the texture has to coincide with the actual geometrical variation of an organ surface. An example of this is the follicles which can be found on the surface of ovaries. A straightforward approach would be to let the user mark those areas where different texture components should be blended. For the blob-shaped follicles, this can be easily done. Figure 3.27 depicts the addition of follicle textures onto an organ surface covered with a synthesized homogeneous texture. In this special case, the follicle texture was not synthesized, but cropped from the original image. While this basic approach leads to acceptable results, it is not appropriate for the scene generation process, since it involves too much manual tuning and image processing.

3.8 Texture Mapping

In the previous sections the generation of textures for surgical scenes has been discussed. What remains to be done is the mapping of those textures to object geometries. If the texture is stored in 3D-space as a trivariate scalar or vector-valued function of indexed or RGB colors respectively, then the surface mapping is trivial, since the texture values can be directly read at the vertex coordinates. No seams or distortions are introduced with this approach. However, 3D texturing is only viable for homogeneous, isotropic patterns. Moreover, the resolution is rather small due to the currently still limited texture memory. Also, the synthesis of 3D texture blocks takes considerably longer than that of their 2D counterparts, thus hampering interactivity of scene generation. If the texture is built as a bivariate field, then a bijective mapping function has to be determined, relating 3D points on a mesh surface to the parametric 2D texture space. This step can usually not be carried out without introducing errors.

A mapping M between two surfaces S_1 and S_2 is called isometric, if the geodesic distance d_{geo} between arbitrary points is maintained:

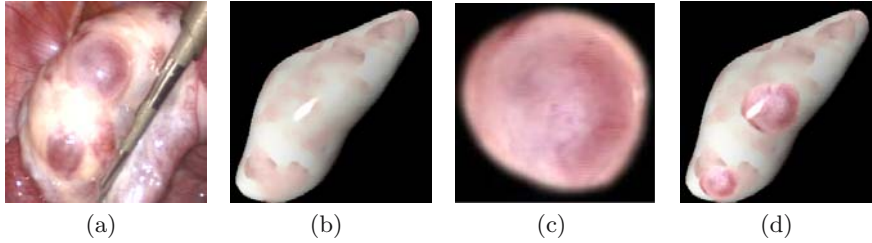<div align="center">(a)   (b)   (c)   (d)</div>

Figure 3.27. Addition of special texture patterns. (a) Follicles on real ovary, (b) Virtual ovary covered with base texture, (c) Cropped follicle pattern, (d) Follicle pattern embedded into base texture

$$\forall \mathbf{p}_i, \mathbf{p}_j \in S_1 : d_{geo}(\mathbf{p}_i, \mathbf{p}_j) = d_{geo}\big(M(\mathbf{p}_i), M(\mathbf{p}_j)\big) . \qquad (3.50)$$

In this case, the surface S_1 is called developable. Gaussian curvature at a point $\mathbf{p}$ on surface S is defined as

$$K = \frac{1}{R_1 R_2} \qquad (3.51)$$

where R_1 and R_2 are the largest and the smallest principle radii of curvature of the respective osculating circles. A sphere has, for instance, a constant positive Gaussian curvature, while that of a cylinder is everywhere zero. Carl Friedrich Gauss stated with his *Theorema Egregium* that an isometric mapping of surfaces with unequal Gaussian curvatures K is not possible [96]. Thus distortions are inevitable when mapping from 2D texture patches with zero curvature to 3D surfaces with non-zero curvature. So a method has to be found to minimize these distortions.

It should be noted that to avoid this problem, one could also directly synthesize textures on the 3D surfaces. Methods to extend the approach discussed in Section 3.6.3 have been presented in [173]. The authors propose techniques to locally project the 3D surface to a tangent plane, which is only successful when the local curvature is not too high. Also, MRF-based synthesis directly on surfaces has been suggested for homogeneous patterns [271, 291, 303], as well as textures with local variations [306]. These approaches require the warping of the spatial neighborhood function over the surface to approximate the mesh curvature. Therefore, generally only relatively small texture structures can be reliably applied to a 3D surface with these methods. Moreover, user interaction is needed to define a vector field over the surface in order to provide a consistent orientation for the texture pattern. Finally, these types of methods do not allow interactive transformations of texture over the 3D surface.

For completeness, it also has to be mentioned that the previously discussed patch-based texture generation strategies can also be directly applied to surfaces. The concept of directly pasting patches onto meshes was initially

introduced in [219]. Refined versions targeting real-time synthesis were later reported in [248, 176]. As in the 2D process, texture discontinuities along seams can appear, which have to be minimized with boundary matching or blending strategies. A drawback also of these approaches is the limited inter-activity, due to the necessary recalculations of texture patches.

While direct texture generation on surfaces is a promising alternative, we have decided to separate the synthesis and surface mapping steps. One reason is the reuse of generated textures, which are stored in a database. A direct mapping strategy would require a resynthesis of textures for each new mesh. Nevertheless, some geometric information of the object meshes still has to be taken into account in the texturing process, as previously exemplified with the mapping of follicle patterns. In the following, the approach followed in the scene generation process will be discussed. It contains two major steps aiming at distortion minimization; namely, mesh cutting along seams and optimal parameterization.

3.8.1 Mesh Cutting

For texture mapping, the object meshes created in the previous chapter have to be parameterized (an extensive overview of current techniques can be found in [86]). Our meshes are represented as piecewise linear triangular surfaces $S_\mathcal{T}$, defined by a set of triangles $\mathcal{T} = \{T_1, \ldots, T_N\}$. For the parameterization, a piecewise linear mapping $f : S_\mathcal{T} \rightarrow S^*$ of the surface $S_\mathcal{T} \in I\!\!R^3$ into the planar domain $S^* \in I\!\!R^2$ has to be found. As discussed above, distortions are usually introduced in this step. A mapping can either be conformal, i.e., angle-preserving; or equiareal, i.e., area-preserving. An isometric mapping is conformal and equiareal; however, since for our meshes such a mapping cannot be found, we minimize a combination of angle and area distortion.

A surface can be mapped to a plane only if it is homomorphic to a disk. Thus a number of cases can be encountered where a surface has to be cut before the mapping step. The genus G of a closed surface is defined by the Euler-Poincare formula:

$$G = \frac{1}{2}E - V - F - B + 2 \tag{3.52}$$

where V, E, and F are the number of mesh vertices, edges, and faces, respectively, and B the number of boundary loops. The genus of a surface can be regarded as the number of holes or handles in a 2-manifold mesh. Surfaces with non-zero genus are not homomorphic to a disk, and thus have to be cut to reduce the genus. The same is true for a manifold mesh with a boundary. If interior holes are present, then these surfaces also have to be separated. Apart from this, the introduction of additional seams by the subdivision of an already disk-like mesh can be applied to further reduce distortion. For the scene generation process, the mesh cutting can be performed according to [235]. Surfaces are separated along existing edges according to two quality
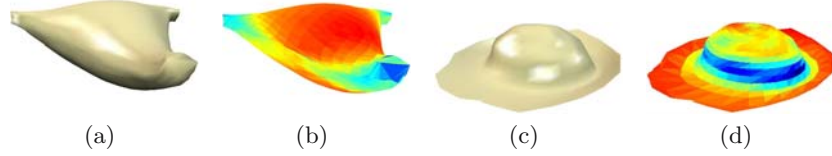
(a)   (b)   (c)   (d)

Figure 3.28. Relative visibility of triangle mesh faces (low=blue, high=red). (a) Uterine cavity surface mesh, (b) Inside visibility measure, (c) Myoma surface mesh, (d) Outside visibility measure

metrics. Firstly, a visibility measure for mesh edges is calculated by rendering the scene from different viewpoints and marking the visible elements. Secondly, the distortion of mesh nodes is determined by estimating the Gaussian curvature. The 3D surface is then separated along its nodes and seams with high distortion, but minimal visibility.

Visibility Measure

In order to obtain edge visibility $V(e)$, a graphics hardware-based approach can be followed. The mesh is rendered orthographically either from multiple viewing positions uniformly distributed on a bounding half-sphere, or into multiple viewing directions from inside an object. This differentiation has to be made, since the pathology meshes will be visible from the outside, whereas the cavity surface will be seen from the inside. By assigning a unique color to each polygon, the visibility of a face can be determined by the presence of these colors in the rendered image. The edge visibility can then be obtained by averaging the visibility of the adjacent faces. The relative visibility of the faces of two example meshes are shown in Figure 3.28. It can be seen that the tubal orifices in the uterine cavity have a low visibility from the inside. Similarly, the visibility of the myoma stem region is reduced, since it is partly covered by the surrounding cavity submesh.

Distortion Measure

In the next step, the distortion at the mesh vertices has to be determined. This can be computed according to an approximation of the local Gaussian curvature. To this end a spherical region around a specific vertex $\mathbf{v}_i$ is considered. The radius of the former is defined according to:

$$R_i = k \max_{\mathbf{v}_j \in \mathcal{N}_i} \|\mathbf{v}_i - \mathbf{v}_j\| \qquad (3.53)$$

where $\mathcal{N}_i$ is the set of nodes adjacent to $\mathbf{v}_i$, and k is a scaling factor. Based on this, we define a local subset of triangles:
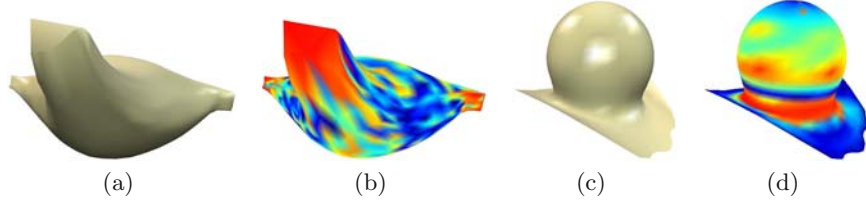
Figure 3.29. Relative distortion at triangle mesh vertices (low=blue, high=red). (a) Uterine cavity surface mesh, (b) Distortion measure, (c) Smooth myoma mesh, (d) Distortion measure

$$\tilde{M}_k(\mathbf{v}_i) = \begin{cases} T_j & | & \forall \mathbf{v}_j \in T_j : \|\mathbf{v}_i - \mathbf{v}_j\| < R_i, & \text{if} \quad k > 0 \\ T_j & | & \mathbf{v}_i \in T_j, & \text{if} \quad k = 0 \,. \end{cases} \qquad (3.54)$$

Thus the subset is composed of all triangles falling completely into the spherical region. The boundary Γ_k of the local subset $M_k(\mathbf{v}_i)$ is piecewise linear. By considering a new set $\hat{\mathcal{N}}_l$ of triangles $\hat{T}_j = \{\mathbf{v}_i, \mathbf{v}^i_{\Gamma_k}, \mathbf{v}^{i\oplus 1}_{\Gamma_k}\}$, which are formed by the center vertex and two consecutive vertices on the boundary, we can define the distortion measure:

$$\Psi'_k(\mathbf{v}_i) = 1 - \frac{\sum_{|\hat{\mathcal{N}}_l|} \alpha_l}{2\pi} \qquad (3.55)$$

where α_l are the angles at vertex $\mathbf{v}_i$ in the new triangles. In order to accommodate for different mesh resolutions, k has to be varied in certain bounds. Unfortunately, the distortion metric is quite sensitive to these variations. Therefore, it is more appropriate to determine the distortions according to

$$\Psi_k(\mathbf{v}_i) = \max_{0 \le q \le k} \Psi'_q(\mathbf{v}_i) \,. \qquad (3.56)$$

It should be noted that at saddle points the metric can become negative. Moreover, if the sum of the angles at a mesh boundary vertex is less than 2π, then it is locally developable and the distortion is zero. Figure 3.29 depicts two examples of the measure. The models are colored according to metric values at the vertices. High distortion is present due to the local mesh curvature at the orifices of the uterine cavity, or at the stem of the myoma.

Optimal Seam Selection

Based on the metrics defined above, seams can now be laid in order to reduce overall mesh distortion. An optimal cut path along the mesh edges has to be found, which minimizes distortion and seam visibility. To this end, a solution to a *Prize-collecting Steiner tree* has to be found. With $G = (V, E)$ being an undirected graph with associated vertex- and edge-weights, the Steiner
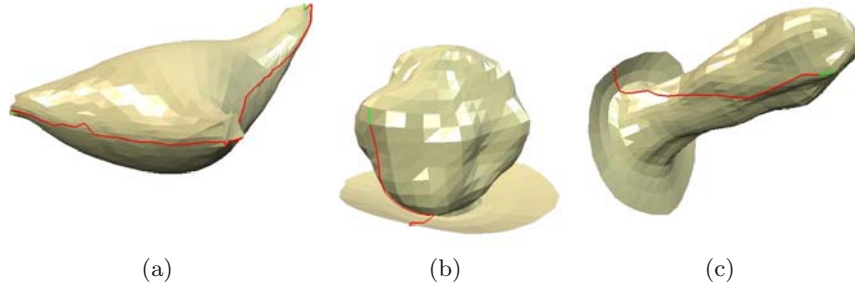
(a)   (b)   (c)

Figure 3.30. Seams shown in red on different meshes, connecting high distortion vertices. (a) Seam on uterine cavity mesh, (b) Seam on myoma mesh, (c) Seam on polyp mesh

problem consists in finding a connected subgraph $T = (V_T, E_T) \subset G, V_T \subseteq V, E_T \subseteq E$, that minimizes the objective function

$$c(T) = \sum_{v \notin V_T} \Psi(v) + \sum_{e \in E_T} \left(V(e) \cdot |e| \right). \qquad (3.57)$$

This problem is known to be NP-complete. An approximation of the solution is obtained following a two-stage process. First, vertices V_T with distortion above a defined threshold – the *terminals* – are selected. The threshold can for instance be set to a percentage of the overall mesh distortion or be user-specified. In order to force the seams to go through less visible areas, the selection can be influenced by considering distortion scaled with vertex visibility $\Psi(v)/V(v)$. In the next step, a Steiner tree connecting the terminals is determined.

A heuristic algorithm using front propagation is used to approximate the result within polynomial time. Starting at each terminal of V_T, a front is grown along the edge with minimal cost. If two fronts meet, their paths are combined, and the resulting subtree is added to the approximate Steiner tree. This is continued until all fronts have merged. The edge cost defined in Equation 3.57 can also be extended by introducing an enhanced edge distortion measure:

$$\sum_{e \in E_T} \left((V(e) \cdot |e|)(1 - \Psi(e)) \right). \qquad (3.58)$$

This results in more cuts being placed along edges of high curvature, where texture artifacts are less visible. Figure 3.30 depicts seams obtained for different meshes. Only a small number of vertices with high distortion were selected as terminals.

3.8.2 Mesh Parameterization

After mesh distortion – as well as possibly mesh genus – have been reduced by introducing additional seams, the actual parameterization step can be carried out, i.e., the mesh is flattened into the 2D plane.

Numerous methods for this step have been suggested in the past. One of the first, described in [272], used graph theory, providing a barycentric mapping theorem for embedding a planar graph. Forces on the mesh vertices are minimized by placing each vertex at the barycenter of its neighbors. This approach was later extended in [85], where shape-preserving weights were introduced. This avoids triangle flips in the presence of obtuse angles, guaranteeing bijective mapping for fixed, convex boundaries. An approach using a discrete approximation of continuous harmonic maps is discussed in [78]. The authors generate parameterizations while minimizing angular distortion with the discrete maps. However, in some cases inverted elements can be generated. In [227] the parameterization problem is considered as an optimal mapping of a signal to a surface. A nonlinear stretch metric derived from Taylor expansions of the signal error is used to reduce distortion. Moreover, to speed up the process, a hierarchical approach is applied. All these methods require a predefined fixed, convex boundary in the parameterization domain. Thus, the boundary mapping has to be specified in advance, preferably to a convex polygon. This also denotes selecting boundary shape and node distribution.

Unfortunately, fixed boundary methods typically introduce additional distortions. This can be avoided with free boundary methods. An additional advantage of these techniques is the usually reduced number of seams required in the mesh. An approach not requiring fixed boundaries has been discussed in [121]. The authors use a *most isometric parameterizations* (MIPS) method, minimizing a nonlinear deformation functional of the first fundamental form of the mapping. However, their method is limited to rather simple cases. In [158], the *least squares conformal maps* (LSCM) technique is proposed, which also uses an adaptive boundary. An error metric based on the discretization of Cauchy-Riemann equations is calculated for constructing free-boundary maps. Unfortunately, bijective mapping is not guaranteed, since folded triangles may appear. Another technique performing free-boundary parameterizationis *angle-based flattening* (ABF) [236]. A functional is minimized according to differences between 2D and 3D mesh angles. This robust and efficient method generates provable conformal mappings with low stretch and no flipped triangles. It is well-suited for the mesh parameterization in the scene generation process and will be discussed in detail below.

Angle-Based Flattening

The angle-based parameterization approach is based on the observation that a planar triangular mesh is defined by the angles within each triangle, except for global scaling, rotation, and translation. An optimization problem is defined,

where a set of constraints guarantees the validity of the flattened mesh. The objective function to be minimized is given by

$$f(\alpha) = \sum_{t \in T} \sum_{k=1}^{3} \frac{1}{w_k^t} (\alpha_k^t - \hat{\alpha}_k^t)^2 \qquad (3.59)$$

where α_k^t are the three unknown planar angles of triangle t in counterclockwise order; and w_k^t are weights, which are usually set to $(\hat{\alpha}_k^t)^{-2}$ to consider relative distortion. The optimal angles $\hat{\alpha}_k^t$ are derived from the 3D mesh angles β_k^t according to

$$\hat{\alpha}_k^t = \begin{cases} 2\pi \cdot \beta_k^t \cdot (\sum_l \tilde{\beta}_l^t)^{-1} & | \quad \mathbf{v}_k^t \in V \backslash V_\Gamma \\ \beta_k^t & | \quad \mathbf{v}_k^t \in V_\Gamma \end{cases} \qquad (3.60)$$

where $\mathbf{v}_k^t$ is the vertex at angle k of triangle t, V_Γ is the subset of boundary vertices, and $\tilde{\beta}_l^t$ are all angles around vertex $\mathbf{v}_k^t$. The unknown angles have to fulfill a number of constraints. Each triangle $t \in T$ has to be valid, thus requiring

$$g_1(t) = \sum_{i=1}^{3} \alpha_i^t - \pi \overset{!}{=} 0 \,. \qquad (3.61)$$

For interior nodes $\mathbf{v} \in V \backslash V_\Gamma$ mesh planarity has to be ensured.

$$g_2(\mathbf{v}) = \sum_{S_\mathbf{v}} \alpha_l^t - 2\pi \overset{!}{=} 0 \qquad (3.62)$$

where $S_\mathbf{v}$ is the set of angles adjacent to vertex $\mathbf{v}$. Finally, edges shared by two triangles should be of equal length. Thus, for each internal vertex

$$g_3^*(\mathbf{v}) = \prod_{S_\mathbf{v}} \frac{\sin(\alpha_{k \oplus 1}^t)}{\sin(\alpha_{k \ominus 1}^t)} \overset{!}{=} 1 \qquad (3.63)$$

where $k \oplus 1$ and $k \ominus 1$ denote in the current triangle the index of the next and previous angle from the one at $\mathbf{v}$, respectively. As indicated in [305], this equation can be rewritten by applying the log function, in order to obtain a diagonal Hessian matrix in the Lagrangian optimization step.

$$g_3(\mathbf{v}) = \sum_{S_\mathbf{v}} \log \left(\sin(\alpha_{k \oplus 1}^t) \right) - \log \left(\sin(\alpha_{k \ominus 1}^t) \right) \overset{!}{=} 0 \,. \qquad (3.64)$$

The constrained minimization problem can be solved using Lagrangian multipliers, giving the modified objective function

$$\begin{aligned} L(x) &= L(\alpha, \lambda_1, \lambda_2, \lambda_3) \\ &= f(\alpha) + \sum_t \lambda_1^t g_1(t) + \sum_\mathbf{v} \lambda_2^\mathbf{v} g_2(\mathbf{v}) + \sum_\mathbf{v} \lambda_3^\mathbf{v} g_3(\mathbf{v}) \,. \end{aligned} \qquad (3.65)$$

(a)   (b)   (c)   (d)
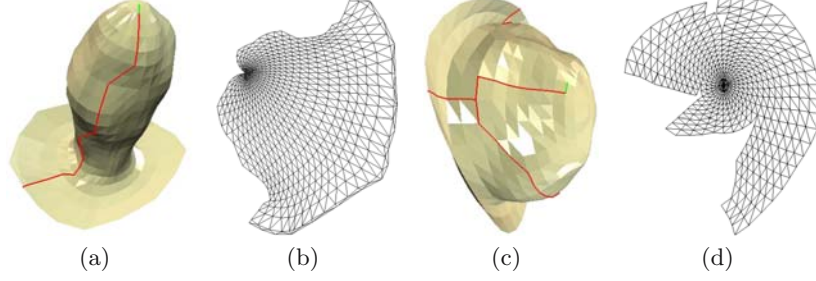
Figure 3.31. Parameterization by angle-based flattening. (a) Polyp mesh with cut seams, (b) Flattened mesh, (c) Myoma mesh with cut seams, (d) Flattened mesh

Inequality constraints can be reformulated as equality constraints using an active set strategy. Thus, a solution is obtained with a standard Newton iteration

$$x_{n+1} = x_n - \nabla^2 L^{-1}(x_n) \cdot \nabla L(x_n) \,. \tag{3.66}$$

For an initial guess the optimal angles $\hat{\alpha}_k^t$ can be selected. Due to the discussed matrix variation, the Hessian is diagonal, but can become ill-conditioned. Using a line search algorithm and a backtracking scheme alleviates this problem.

2D Mesh Retrieval

After solving for the 2D planar angles, the actual vertices need to be reconstructed in the plane. Using a front propagation method, visiting one vertex at a time can quickly become unstable from a lack of numerical precision. Instead, the conversion problem can be formulated as a global linear system, thus enabling the simultaneous computation of the 2D vertices. Given a triangle $t = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$, then the edge $\overline{\mathbf{p}_1\mathbf{p}_3}$ is given by:

$$\overline{\mathbf{p}_1\mathbf{p}_3} = M^t \cdot \overline{\mathbf{p}_1\mathbf{p}_2} = \frac{\sin(\alpha_2^t)}{\sin(\alpha_3^t)} \begin{pmatrix} \cos(\alpha_1^t) \ \sin(\alpha_1^t) \\ -\sin(\alpha_1^t) \ \cos(\alpha_1^t) \end{pmatrix} \overline{\mathbf{p}_1\mathbf{p}_2} \tag{3.67}$$

where α_i^t is the angle at vertex i of triangle t. Thus, the third vertex is uniquely defined with respect to the position of the two other vertices and the angles.So to obtain the positions, a solution in the least-squares sense has to be found for:

$$\min_{\mathbf{p}_i} \sum_{t=(\mathbf{p}_i,\mathbf{p}_{i+1},\mathbf{p}_{i+2})\in T} \|M^t(\mathbf{p}_i - \mathbf{p}_{i+1}) + \mathbf{p}_{i+2} - \mathbf{p}_{i+1}\|^2 \,. \tag{3.68}$$
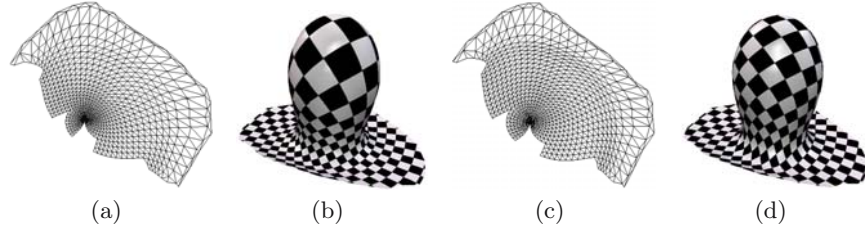
| (a) | (b) | (c) | (d) |

Figure 3.32. Adapted parameterization to reduce linear distortions. (a) Flattened mesh after standard ABF, (b) Distorted checkerboard texture, (c) Improved parameterization, (d) Enhanced length preservation

It should be noted that the mesh is defined except for translation, rotation, and scaling. To eliminate these degrees of freedom, two vertices sharing an edge can be fixed, e.g., by setting P_0 and P_1 to $(0,0)$ and $(1,0)$. This minimization problem is well defined and has a unique minimum. The system can for instance be solved with a SuperLU direct solver, thus finally providing the parameterization for the 3D mesh, which can be used for texturing. Examples of the outcome of the angle-based flattening procedure are visualized in Figure 3.31.

Reduction of Length Distortion

In the process carried out so far, a parameterization minimizing angular distortion has been obtained. However, due to the angle-based formulation, edge lengths are not preserved, thus introducing linear distortions. The mesh smoothing algorithm suggested in [234] can be applied to alleviate this situation.

The first step is to define a linear approximate distortion function ρ based on the erroneous edge lengths. With average distortion $\tilde{\rho}_i$ at mesh vertex i, the distortion at an arbitrary point $\mathbf{x}$ inside a triangle of the parameterization mesh with barycentric coordinates (u, v, w) can be defined as:

$$\rho(\mathbf{x}) = \tilde{\rho}_1 u + \tilde{\rho}_2 v + \tilde{\rho}_3 w \tag{3.69}$$

where the average distortion is the mean of the edge length change ratios $\rho^* = \|e_2 D\| \cdot \|e_3 D\|^{-1}$ of all adjacent edges. In order to cancel out the length distortion, the inverse mapping ρ^{-1} has to be determined. To this end, a uniform, regular grid can be adapted using Laplacian smoothing to conform to the distortion function. After obtaining the adapted mesh, the actual mapping back to the original, regular mesh approximates ρ^{-1}, and can thus be applied to the initial parameterization. The effect of this step is depicted in Figure 3.32. In order to better visualize the change, a regular pattern has been applied to the geometry. It should be noted, however, that the visual effect is less noticeable for more homogeneous, stochastic organ textures.
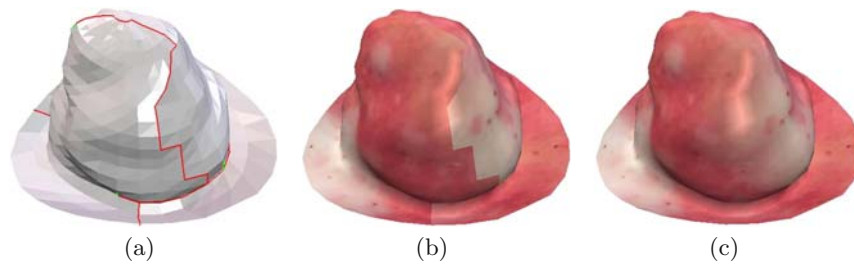
(a)   (b)   (c)

Figure 3.33. Blending strategy to avoid visual artifacts. (a) Myoma mesh with cut seams, (b) Noticeable texture discontinuities across seam, (c) Artifact removal via alpha blending

Seam Blending

Thereafter, the texture can be projected back to the 3D surface by inverse mapping. Due to the selected approach, texture discontinuities exist along the cut seams of the mesh. Although number, length, and visibility of the seams have been minimized, these discontinuities still create unwanted visual artifacts. To reduce these artifacts, hardware-based blending can be applied. Given a 2D planar mesh of a 3D surface, and a table of matching edges along the seams, duplicate triangles along the seams can be introduced. These contain the extrapolated texture coordinates from across the seam. Vertices directly on the seam are given an alpha value of 0.5, while the rest are incrementally decreased as the overlaid triangles progress away from the edge. Alpha blending can then easily be performed with standard graphics libraries. The result of this step is shown in Figure 3.33. It should be noted that this example shows an extreme case of seam discontinuity. Usually the seams are less noticeable due to more homogeneous textures and the visibility reduction.

3.9 Modelling Examples

The methods described above have been used either in the laparoscopic or the hysteroscopic simulator projects. MRF-based texture synthesis and ABF-based parameterization have been integrated into the scenario definition process for hysteroscopic training scenes. In Figure 3.34 several examples of texture mapping onto meshes relevant for laparoscopy as well as hysteroscopy are shown. In a hysteroscopy scene, a number of different tissue structures are present. Healthy anatomy as well as neoplasms exist, which usually have different visual appearances due to the varying genesis of these objects. Therefore, the texturing processes are carried out separately for the object submeshes. Since texture discontinuities similar to the ones discussed above will appear, an overlap region is included for each object mesh. By using several triangle strips around the mesh boundaries, the blending across different textures
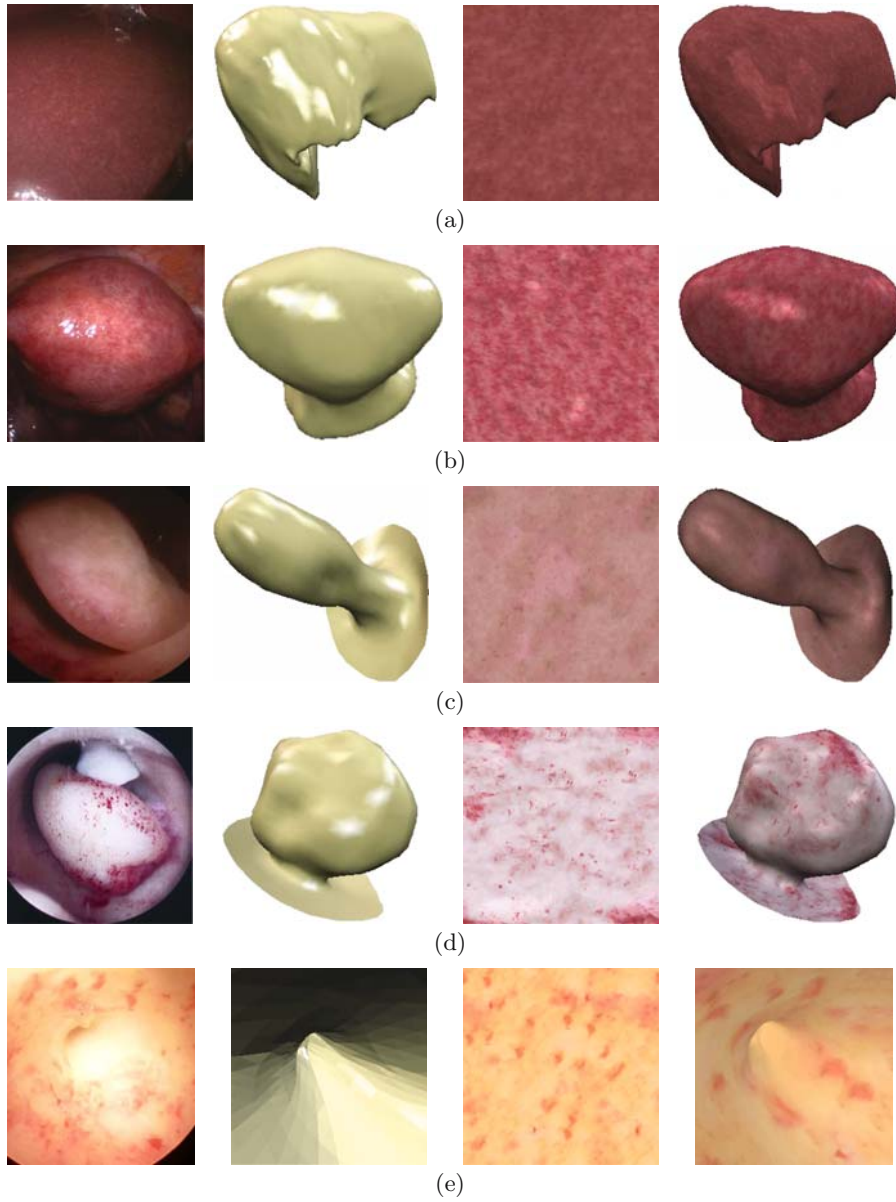
(a)

(b)

(c)

(d)

(e)

Figure 3.34. Examples of textured meshes showing source image, object mesh geometry, synthesized texture, and final result. (a) Liver, (b) Uterus, (c) Polyp, (d) Myoma, (e) Endometrium around tubal ostium
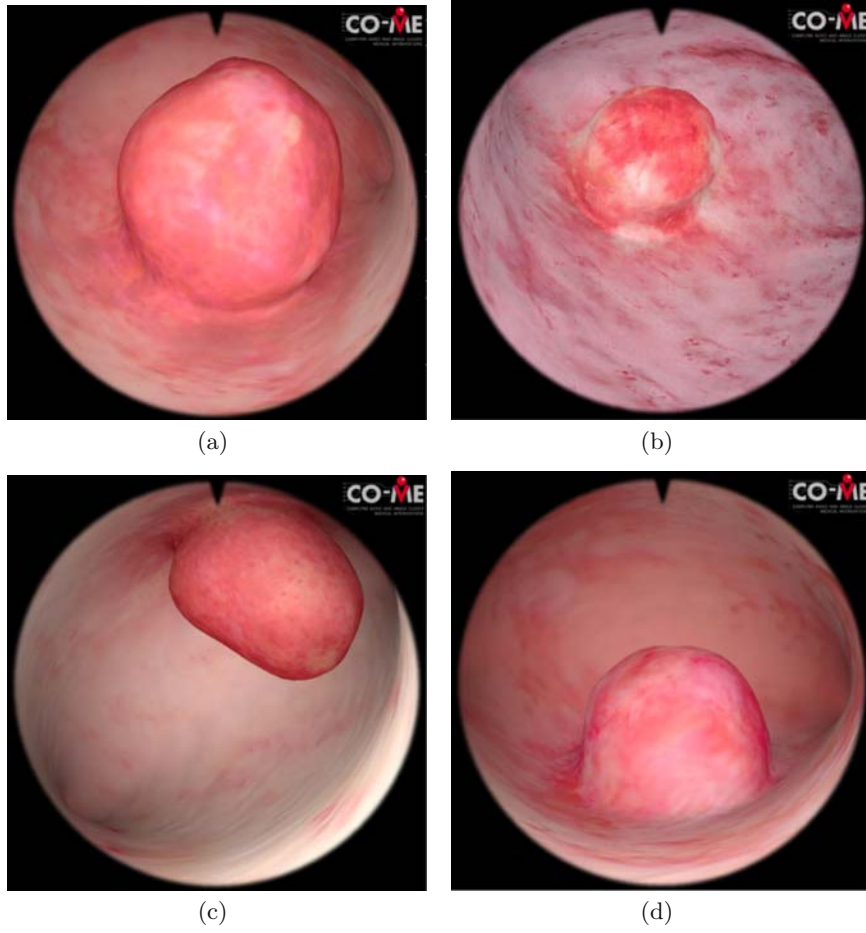
Figure 3.35. Complete hysteroscopy scene examples

can be improved. Several complete hysteroscopy scenes textured with the discussed method are presented in Figure 3.35.