

A Framework for Coherent Texturing in Surgical Simulators

Rupert Paget, Matthias Harders, and Gábor Székely
Swiss Federal Institute of Technology
Computer Vision Lab
ETH Zentrum, CH-8092 Zürich, Switzerland
{rpaget, mharders, szekely}@vision.ee.ethz.ch

Abstract

An often neglected key element for surgical simulators is the ability to generate new, individual training scenes for every session. This is necessary to avoid the adaptation of the trainee to the presented case. To this end, we have developed a coherent scenario generation process, including organ geometries, deformation parameters, and visual appearance. In this paper we describe the framework developed to address the latter element. Our system enables automatic generation of textures for surgical simulation based on in-vivo images.

1 Introduction

A central driving force in our current research is the development of a highly realistic simulator for hysteroscopic interventions [2]. In contrast to existing systems and products [11, 16, 14], we strive to achieve the highest possible realism - key idea being a reference system, which can serve as a Golden Standard for surgical performance. With this strategy we try to address the open question of the necessary level of realism for a desired training effect. To this end, we will observe the effect of changing system fidelity on the skill acquisition process.

A key point in this framework is the instantiation of a coherent surgical scene. Just like in a flight simulation, where different weather conditions, airports, system malfunctions, etc. can be defined, surgical simulation also needs the same breadth of configurable training conditions. In current simulators this point is usually neglected. Single static organ models are used, homogeneous, manually selected tissue parameters are applied, and visual appearance is limited to the predefined scene.

In [27] we have introduced a coherent scene generation process, which can be used to create variable sce-

narios, reflecting differences in individual patients. The system addresses variability of healthy organs, growth simulation for pathologies, tissue vascularization, and automatic adaptation of deformation parameters. A further link in this chain is the visual appearance of geometries in the scene. This denotes the generation of realistic textures, as well as obtaining appropriate texture coordinates. In this paper we described the fully automatic texturing framework, which addresses all the requirements related to the problem area at hand.

The flow of the process of our framework is depicted in Figure 1. The first step is the acquisition of in-vivo images to form a ground-truth database. Next, tileable, variable textures are created from the in-vivo images in a texture synthesis step. Thereafter, the texture is mapped to the 3D mesh geometry. This is done by mesh parameterization, which takes into account visibility of seams and distortion reduction. Finally, textures are blended across the seams and the junctions between different objects, e.g. pathologies and healthy tissue, to reduce boundary artifacts.

2 Previous Work

Since generation of variable training scenes in surgical simulators has only found very limited attention in the past, synthesis of variable textures also has not been extensively covered in this area.

In [9] 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 method has been applied in a simulator for minimally invasive neurosurgery. Real images acquired during endoscopy were used to manually generate the textures. A different strategy using texture samples and avoiding distortion, discontinuity and repetitiveness is presented in [19]. They used their method to apply tex-

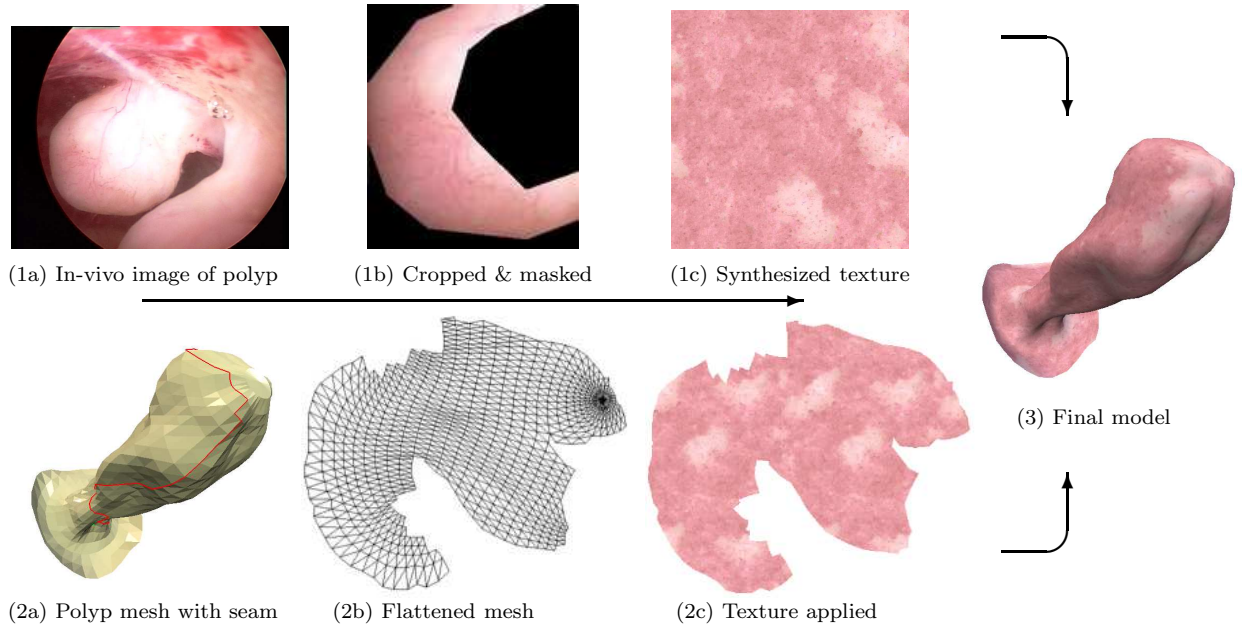


Figure 1: Model texturing process.

ture samples to a 3D mesh of a liver, however, it does not become clear, how they obtained the tissue surface samples. This work has later on been extended in [18] by adding dynamic effects, for instance cauterization or blood drops. A more complete coverage of the problem of texture generation in surgical simulation has been presented in [15]. Organ specific base textures are automatically computed by means of a texture analysis/synthesis process. Moreover, small features of tissue surfaces, which are not captured in the analysis/synthesis step, can be added with a procedural texturing approach. Related to the problem at hand, is another recent trend in surgical simulation, which examines the use of image-based rendering for realistic visualization. For instance in [5], real images are combined with artificial specular reflections, modulated by a set of reflectance maps, to simulate different views of the endoscopic camera.

Without focus on the special needs of the application area, further work has been carried out in general texture generation. One approach is direct texture synthesis on the 3D surface [30, 33, 7, 34, 36]. These approaches used the Markov random field method of texture synthesis [20, 3, 32]. This requires the warping of the spatial neighborhood function over the surface to approximate 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 polarize the texture. In any case, these types of methods are by no means interactive when it comes to modifying the texture over the 3D surface.


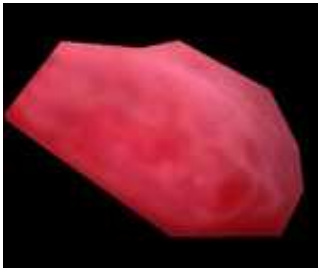
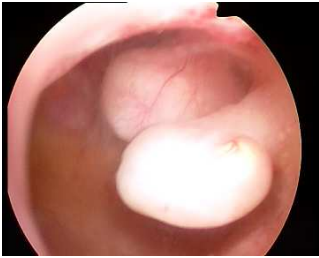

An alternative approach is to paste patches of the texture directly onto the surface [19, 21, 28, 13]. While these methods perform better than direct synthesis, they do suffer from texture discontinuities along seams (except for [19]). However, these can be minimized with boundary matching or blending, or a combination of both. Unfortunately, a drawback of these approaches is the limited interactivity, due to the necessary recalculations of texture patches.

3 Texturing Framework

3.1 In-Vivo Data Acquisition

While in some application areas it is possible to completely synthesize artificial textures, this is not a viable approach in surgical simulation. A ground truth covering several aspects of visual appearance of organs has to be obtained, which serves as input to the texture generation chain. For this reason, 10 hours of in-vivo video taken during various hysteroscopies were acquired. Based on this material, a texture database was created. The images were selected to represent different structures in the uterine cavity. Usually, high

Table 1: Excerpt of in-vivo image database.

Original image	Cropped & masked image	Metrics
		<ul style="list-style-type: none"> • ID: 105 • Medical term: polyp • Description: complete polyp connected to uterus • Age: postmenopausal • Resolution: medium • Size: 374x312 • Focus: excellent
		<ul style="list-style-type: none"> • ID: 142 • Medical term: polyp • Description: polyp • Age: unknown • Resolution: medium • Size: 198x140 • Focus: good

resolution views of characteristic tissues were chosen, which optimally were perpendicular to the surface and without strong camera spotlight effects. We obtained 69 different samples that contained usable textures of various types of tissue. These were then cropped and masked to obtain just the desired surface regions. In Table 1 a few categorized textures of our searchable database are visualized.

3.2 Texture Synthesis

The next step is the synthesis of new textures based on images of the in-vivo database. Since our training scenarios should differ from session to session, this has to be done in a stochastic process. Another requirement of this step is that the synthesized textures are tileable, which will be necessary for the next phase.

Currently existing texture synthesis algorithms can basically be sorted into two categories. Pixel-based approaches and patch-based methods. We decided to use the former, obtaining the required images with a fast non-parametric texture synthesis (FNTPS) approach. It is modified version of the multi-scale texture synthesis algorithm presented in [20]. It is sped up with a neighborhood searching scheme similar to the one proposed in [1]. For the neighborhood cost function we use the

Manhattan distance:

$$\text{Cost}_j = \sum_i^N \begin{cases} |s_i - r_i| \frac{n_i}{T} & \text{if } s_i \text{ is defined} \\ \text{Max} \frac{n_i}{T} & \text{otherwise} \end{cases} \quad (1)$$

where $s_i \in S$ are pixels in source image (s_i may be undefined if it is outside the image, or is masked), and $r_i \in R$ are pixels in the synthetic image. The factor n_i/I equals the number of times the output pixel has been iterated divided by the proposed number of iterations per pixel. If s_i is undefined, then a cost, equal the maximum possible cost, is added to the overall neighborhood cost.

The cost function is calculated over a subset of sampling pixels from the source image. The pixel that returns the lowest neighborhood cost is chosen as the one to transfer its color to the respective pixel in the synthetic image. The iteration count is then incremented for that output pixel. Once all output pixels have reached their proposed number of iterations, the synthesis process stops. The site visitation sequence across the output pixels can be in any random order as describe in [20].

In contrast to [1], the subset we sample from are all pixels with a neighbor of the same color as the respective neighbor of output pixel being iterated. This is a larger subset than proposed by [1], which results in

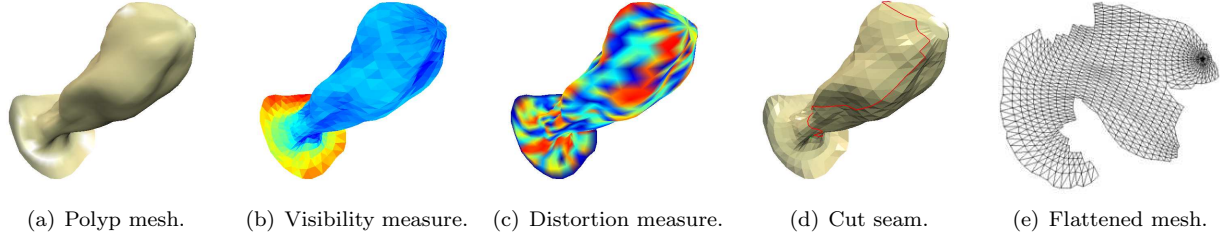


Figure 2: Flattening of 3D mesh for parameterization

a higher synthesis quality, but with only minimal increase in computation time. The perceptual similarity measure between the neighborhoods that was proposed by [1] is also maintained with this approach.

Compared to patch-based algorithms (e.g. [8, 4, 12, 17]), our pixel-based approach produces more stochastic variations in the synthetic texture, while providing comparable results. Finally, due to its pixel-based strategy, our method can easily cope with image masking. Using FNTS, we were able to produce a variety of different tileable tissue textures from our database, as seen in Figure 3.

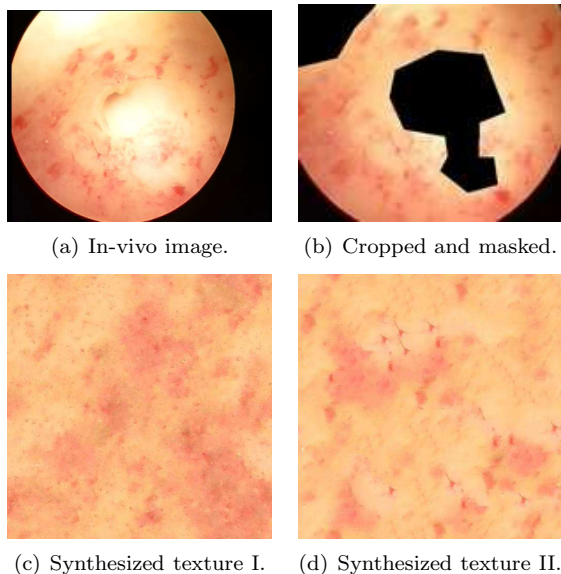


Figure 3: Texture synthesis of in-vivo texture

3.3 Mesh Parameterization

Following the texture synthesis, the 2D image now has to be mapped to the 3D mesh of the organ. The geometry of the surgical scene is generated as part of our framework. Since organs of any two patients will never be alike, we use statistical anatomical models to handle

the variability of healthy human anatomy. Thereafter, different pathological cases are artificially created by growth processes, and added to the scene. These give us triangle surface meshes, to which the synthesized textures are applied. The mesh parameterization is carried out by cutting the 3D mesh and mapping it to a 2D plane. Thereafter, the texture can be projected back to the 3D surface by inverse mapping.

When deciding how to cut the mesh, one first needs to decide which parameterization method they wish to use. Basically mesh parameterizers fall into two categories [6]; ones that require a fixed convex boundary, and those that do not. The ones that allow for a free boundary typically provide reduced distortion and require fewer seams over the mesh. Of these types of methods there are basically two that at least partially guarantee that no triangle flips will occur. There is angle based flattening (ABF) [26, 22], and then there is least squares conformal maps (LSCM) [10], but as discussed in [22], ABF performs a lot better. Alternatively there is the bounded-distortion method [29], which integrates both the cutting and parameterization into one algorithm. However, there is little control over the placement of the seams and there tend to be a lot of them, which makes this method comparable to some of the texture patch pasting methods.

In this work we have chosen to use ABF [26, 22], where the mesh cutting is done according to [24], from which the surface is separated along existing edges. This process is carried out according to two quality metrics. Firstly, a visibility measure for mesh edges is calculated. This is done, by rendering the scene from different viewpoints and marking of visible elements. Secondly, distortion of mesh nodes is calculated by estimating the Gaussian curvature. The 3D surface is then separated along nodes that contribute significantly to the total mesh distortion, and minimally visible seams.

This problem is NP-Complete, however, a viable solution can be obtained with the approximate minimal Steiner tree [25]. A number of alternatives exists to calculate the edge cost in this method. We chose to supplement the calculation with a curvature cost as de-

scribe in [10]. The final cost for each edge is calculated as:

$$e_{cost} = |e|e_{visibility}(1 - e_{curvature})$$

This helps to place cuts along edges of high curvature, in which texture artifacts are less visible.

After cutting, the mesh is ready for flattening and texture parameterization. We applied angle based flattening with the SuperLU method [22]. However in our case we implemented the matrix variation as describe in [35]. This creates a simpler matrix, but it can become ill-conditioned. Employing line searching and backtracing to the Newton-Raphson method does well in circumventing this problem.

After solving for the 2D planer angles, vertices need to be extrapolated to a 2D plane. The traditional method of growing the mesh by using the angles to find the 3rd vertex in each triangle can quickly become unstable from a lack of numerical precision. Therefore we used the global optimizer according to [31]. Finally, since angle based flattening does not preserve edge lengths, we implemented a mesh smoothing algorithm to reduce the length distortion [23]. An example of this process for a mesh of a polyp is shown in Figure 2.

3.4 Texture Blending

In parameterizing the 3D surface mesh, it was cut and flattened. Then, a texture was applied to the flattened mesh, which thereafter was backprojected onto the 3D surface. With this method a texture discontinuity exists along the cut seam 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, we apply an enhancement step based on alpha blending.

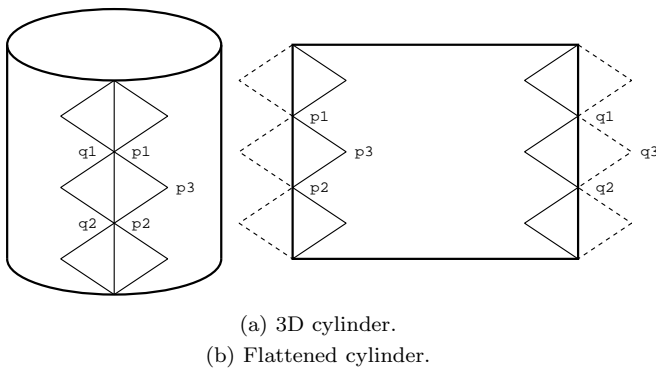
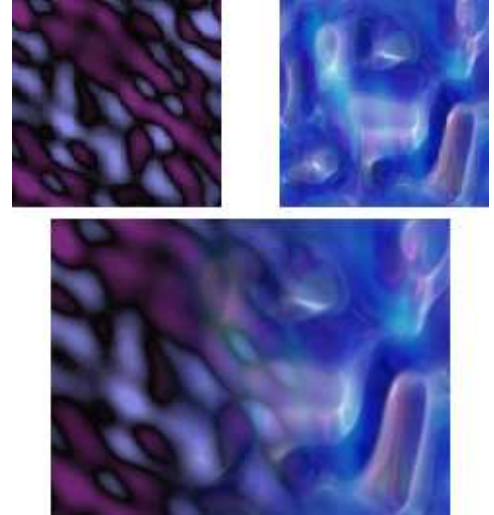
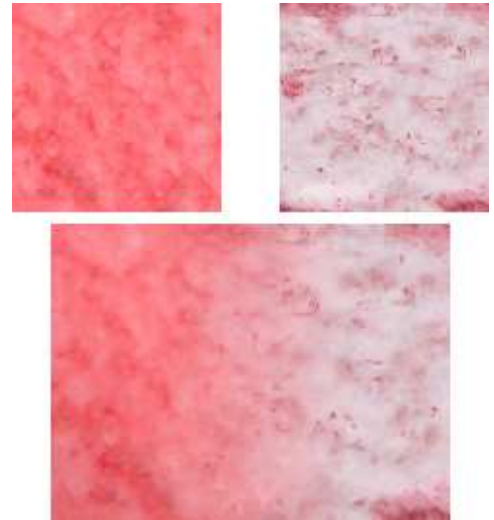


Figure 4: Blending across mesh boundaries.

Given a 2D planar mesh of a 3D surface, and a table of matching edges along the seams, it becomes possible



(a) Artificial textures.

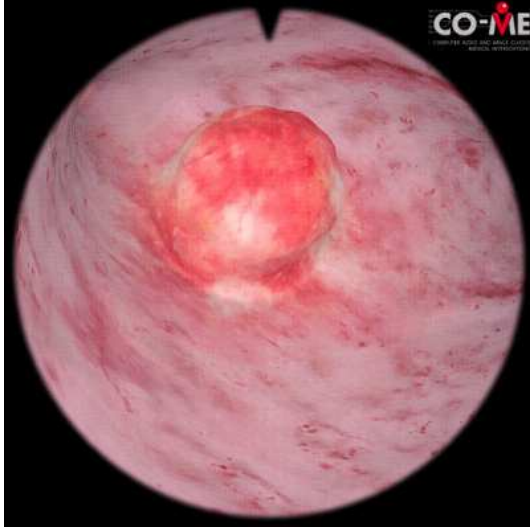


(b) Synthesized organ textures.

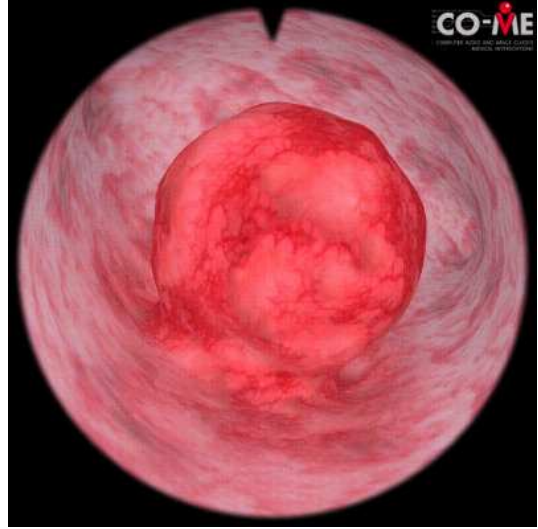
Figure 5: Blending across object boundaries.

to precompute the alpha blending parameters. This is done with duplicate triangles along the seams that 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 decremented as the overlaid triangles progress away from the edge. Alpha blending is then easily performed in OpenGL by setting the material properties of each vertex of the overlaid triangle to the corresponding alpha value.

Consider, for example, Figure 4. Given the texture coordinates $p1$, $p2$, and $p3$ of the current triangle, and the texture coordinates $q1$ and $q2$ of the opposite triangle where the 3D vertices of $p1$ and $p2$ correspond to



(a) Myoma scene.



(b) Another myoma scene.

Figure 6: Similar geometries with different textures.

$q1$ and $q2$ respectively, then the 3rd coordinate $q3$ of the triangle to be overlaid onto the current triangle can be calculated as follows:

$$\begin{aligned} v &= p1 - p2, & u &= q1 - q2, & w &= p3 - p1 \\ vn &= v/|v|, & un &= u/|u| \\ vt[0] &= -vn[1], & vt[1] &= vn[0], & vt[2] &= 0.0 \\ ut[0] &= -un[1], & ut[1] &= un[0], & ut[2] &= 0.0 \\ q3 &= \left(\frac{|u|}{|v|} ((w \cdot vn)un + (w \cdot vt)ut) \right) + q1 \end{aligned}$$

The discussion so far has only focused on texturing of a single mesh. However, in our scene generation, a number of different structures exist, mainly healthy anatomy and neoplasms. In the case of hysteroscopy, we artificially grow myomas and polyps in the uterine cavity. Due to the differing genesis of these objects, they usually have different visual appearance. Therefore, the process above has to be carried out for these objects separately, which makes the handling of the mesh interfaces necessary. To this end, we have developed a blending approach between object seams.

Blending across objects is a lot more straightforward than the one within an object described above. Basically we define our complete scene with a set of submeshes, where each submesh encapsulates the surface covered by just one texture. However for each submesh we also include an overlap region between meshes. These submeshes are then processed individually and texture is applied. The overlap region contains the necessary information to apply alpha blending between objects. It is easy to make this overlap region quite large,

which improves the blending across different textures, as shown in Figure 5.

3.5 Integration into Hysteroscopy Simulator

The texturing framework described above has been integrated into our model generation process. This now enables us to generate varying training scenes, including geometries and visual appearance, as well as vessel structures and tissue parameters. In Figure 6 two examples textured with our method are presented.

4 Conclusion

To acquire surgical skills, it is highly desirable that the training scene is different from session to session - just like in real practice. This includes geometries, as well as visual appearance. To this end, we have developed an automatic method for generating textures for our training scenarios.

We have presented a method for synthesizing 2D textures based on in-vivo images, and mapping these to 3D surfaces of organ geometries. Although texture synthesis is usually performed on a 2D lattice, our approach allows the texturing of 3D surfaces. The method enables the mapping of large scale textures to complex shapes without significant distortions. A minor drawback of the algorithm is the occurrence of discontinuities along seams, however, we alleviate this by blending across the object interfaces.

As the texture mapping is performed via a complete mesh parameterization, the whole texturing framework is interactive. A texture can be scaled, rotated, or translated across the 3D surface interactively. Alternatively the texture can be simply replaced. This all can be done with synchronous blending. The presented texturing framework makes it easy to design and create individual texturing scenarios to meet the needs of the surgical training simulator.

Acknowledgment

The authors would like to thank all members of the hysteroscopy simulator team. This research has been supported by the NCCR Co-Me of the Swiss National Science Foundation.

References

- [1] M. Ashikhmin. Synthesizing natural textures. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226, 2001.
- [2] ACOG Tech. Bulletin. Hysteroscopy. *Int J Gyn. Obstet.*, May 1994. 45(2): 175-180.
- [3] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, volume 2, pages 1033–1038, September 1999.
- [4] A.A. Efros and W.T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001.
- [5] M.A. ElHelw, B.P.L. Lo, A.J. Chung, A. Darzi, and G.-Z. Yang. Photorealistic rendering of large tissue deformation for surgical simulation. In *MICCAI*, volume 2, pages 355–362, 2004.
- [6] M. S. Floater and K. Hormann F N. A. Dodgson. Surface parameterization: a tutorial and survey. In M. S. Floater and M. A. Sabin, editors, *In Advances in Multiresolution for Geometric Modelling*, pages 259–284. Springer, 2004.
- [7] Gabriele Gorla, Victoria Interrante, and Guillermo Sapiro. Growing fitted textures. In *SIGGRAPH 2001 Sketches and Applications*, page 191, August 2001.
- [8] B. Guo, H. Shum, and Y.-Q. Xu. Chaos mosaic: Fast and memory efficient texture synthesis. Technical report, Microsoft Research, 2000.
- [9] V. Leeb, A. Radetzky, and L.M. Auer. Interactive texturing by polyhedron decomposition. In *IEEE Proceedings on Virtual Reality*, pages 165–171, 2001.
- [10] B. Levy, S. Petitjean, N. Ray, and J. Mailliot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 362–371, 2002.
- [11] J.S. Levy. Virtual reality hysteroscopy. *J Am Assoc Gyn. Laparosc.*, 3(4, Suppl.):25–26, 1996.
- [12] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.
- [13] Sebastian Magda and David Kriegman. Fast texture synthesis on arbitrary meshes. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 82–89, 2003.
- [14] Immersion Medical. AccuTouch system. Company webpage (visited Mar 2005).
- [15] V. Meier. *Realistic visualization of abdominal organs and its application in laparoscopic surgery simulation*. PhD thesis, ETH Zurich, 1999.
- [16] K. Montgomery et al. Surgical simulator for hysteroscopy: A case study of visualization in surgical training. In *IEEE Visualization*, 2001.
- [17] Andrew Nealen and Marc Alexa. Hybrid texture synthesis. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 97–105, 2003.
- [18] F. Neyret, R. Heiss, and F. Senegas. Realistic rendering of an organ surface in real-time for laparoscopic surgery simulation. *the Visual Computer*, 18(3):135–149, 2002.
- [19] Fabrice Neyret and Marie-Paule Cani. Pattern-based texturing revisited. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 235–242, 1999.
- [20] R. Paget and D. Longstaff. Texture synthesis via a noncausal nonparametric multiscale Markov random field. *IEEE Transactions on Image Processing*, 7(6):925–931, 1998.

- [21] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 465–470, 2000.
- [22] A. Sheffer, L. Bruno, M. Maxim, and B. Alexander. ABF++ : Fast and Robust Angle Based Flattening. *ACM Transactions on Graphics*, 2004.
- [23] A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Trans. Graph.*, 21(4):874–890, 2002.
- [24] A. Sheffer and J.C. Hart. Seamster: inconspicuous low-distortion texture seam layout. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 291–298, 2002.
- [25] Alla Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, page 61, 2002.
- [26] Alla Sheffer and Eric de Sturler. Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17(3):326–337, 2000.
- [27] R. Sierra, M. Bajka, C. Karadogan, G. Szkely, and M. Harders. Coherent scene generation for surgical simulators. In *Medical Simulation: International Symposium*, pages 221 – 229, 2004.
- [28] Cyril Soler, Marie-Paule Cani, and Alexis Angelidis. Hierarchical pattern mapping. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 673–680, 2002.
- [29] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 355–362, 2002.
- [30] Greg Turk. Texture synthesis on surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 347–354, 2001.
- [31] B. Vallet, B. Levy, and J.-C. Paul. Constrained jacobian parameterization. Technical report, LORIA/INRIA, 2003.
- [32] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 2000, 27th International Conference on Computer Graphics and Interactive Techniques*, pages 479–488, 2000.
- [33] Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 355–360, 2001.
- [34] Lexing Ying, Aaron Hertzmann, Henning Biermann, and Denis Zorin. Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 301–312, 2001.
- [35] R. Zayer, C. Rossl, and H.-P. Seidel. Variations on angle based flattening. In *Proceedings of Multiresolution in Geometric Modelling*, pages 285–296, 2003.
- [36] Jingdan Zhang, Kun Zhou, Luiz Velho, Bain-ing Guo, and Heung-Yeung Shum. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Trans. Graph.*, 22(3):295–302, 2003.