Generating Sub-Resolution Detail in Images and Volumes Using Constrained Texture Synthesis

Lujin Wang Klaus Mueller
Center for Visual Computing, Computer Science, Stony Brook University

ABSTRACT

A common deficiency of discretized datasets is that detail beyond the resolution of the dataset has been irrecoverably lost. This lack of detail becomes immediately apparent once one attempts to zoom into the dataset and only recovers blur. Here, we describe a method that generates the missing detail from any available and plausible high-resolution data, using texture synthesis. Since the detail generation process is guided by the underlying image or volume data and is designed to fill in plausible detail in accordance with the coarse structure and properties of the zoomed-in neighborhood, we refer to our method as constrained texture synthesis. Regular zooms become "semantic zooms", where each level of detail stems from a data source attuned to that resolution. We demonstrate our approach by a medical application – the visualization of a human liver – but its principles readily apply to any scenario, as long as data at all resolutions are available. We will first present a 2D viewing application, called the "virtual microscope", and then extend our technique to 3D volumetric viewing.

CR Categories: I.3.7 [Computer Graphics]: Color, shading, shadowing and texture I.3.3 [Computer Graphics]: Picture/Image Generation

Keywords: texture synthesis, semantic zoom

1 INTRODUCTION

When viewing an image (note, a volume is considered a 3D image for the discussion here) the amount of detail that can be visually explored is fundamentally bounded by the image resolution. Magnification will not extend the amount of visible detail, it will only spread it out in space such that it can be better discerned by the observer. Magnification typically entails some blurring, depending on the quality of the magnification filter used [20]. However, it should be obvious that even with the best filter, pure magnification can not add detail where it has not been sampled before. Therefore, zooming into an image or volume at high magnification factors tends to create a rather boring, non-informative, and non-satisfying viewing experience.

The amount of available detail may be constrained by: (i) economical limits bounding the size and therefore the detail of the image, and/or (ii) technical limits inherent in the image acquisition process. As an example for the latter, optical lenses generally are only able to provide focus within a certain range of scale, while imaging technologies, such as MRI and CT, impose physical limits on the amount of detail they can resolve. Should detail on other scales be desired, alternative lenses or imaging methods, such as optical, confocal, and electron microscopy are required.

e-mail: lujin@cs.sunysb.edu
e-mail: mueller@cs.sunysb.edu

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

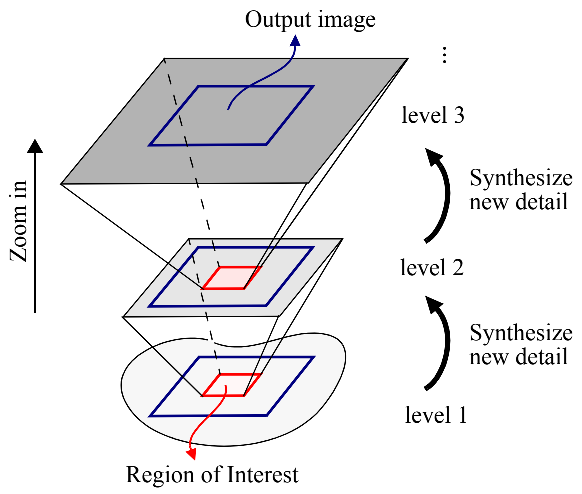


Figure 1: Semantic zooming based on texture synthesis

In computer graphics, texture mapping has long been a method by which interesting detail can be added. However, texture placement is usually guided by geometry, and not by semantic constraints imposed by the image to be enriched. Texture mapping may also cause repetitive tiling artifacts. Texture synthesis has more promise in this respect. For example, Freeman et al. [8] established a database of coarse-fine resolution mappings that they used to add fine detail to magnified images of natural scenes. This fine detail, however, was on the same order of scale as the base image, and only magnifications at the same semantic level of scale were possible.

In this paper, we propose to extend the notion of image-guided detail enhancement to multiple levels of scales. However, we would like to avoid traditional image pyramids where multi-scale detail stems from the repeated smoothing of a single high resolution image. This is because requiring such an image would violate one or both of the constraints mentioned above. Instead, we introduce the notion of semantically constrained multi-scale texture synthesis to facilitate zooms at a virtually infinite number of scales, as long as the corresponding texture data are available (see Fig. 1). Here, the term "semantic zooming" means that the multi-scale detail is not derived from one image to the other via simple filtering, but via different sampling processes tuned to the respective level of scale. An everyday example of semantic zooming [7] is electronic maps, where each level of zoom is an excerpt of a different map, such as country, state, city, neighborhood, etc., bearing a very different style and type of detail.

In contrast to the aforementioned maps, our application does not store complete images at every level. One of our main design goals is to generate the semantic detail at a minimum of memory cost, thus providing a solution that will scale well. Therefore, our system will not yield an accurate multi-scale "map", rather, it will generate something that looks like an accurate multi-scale map,

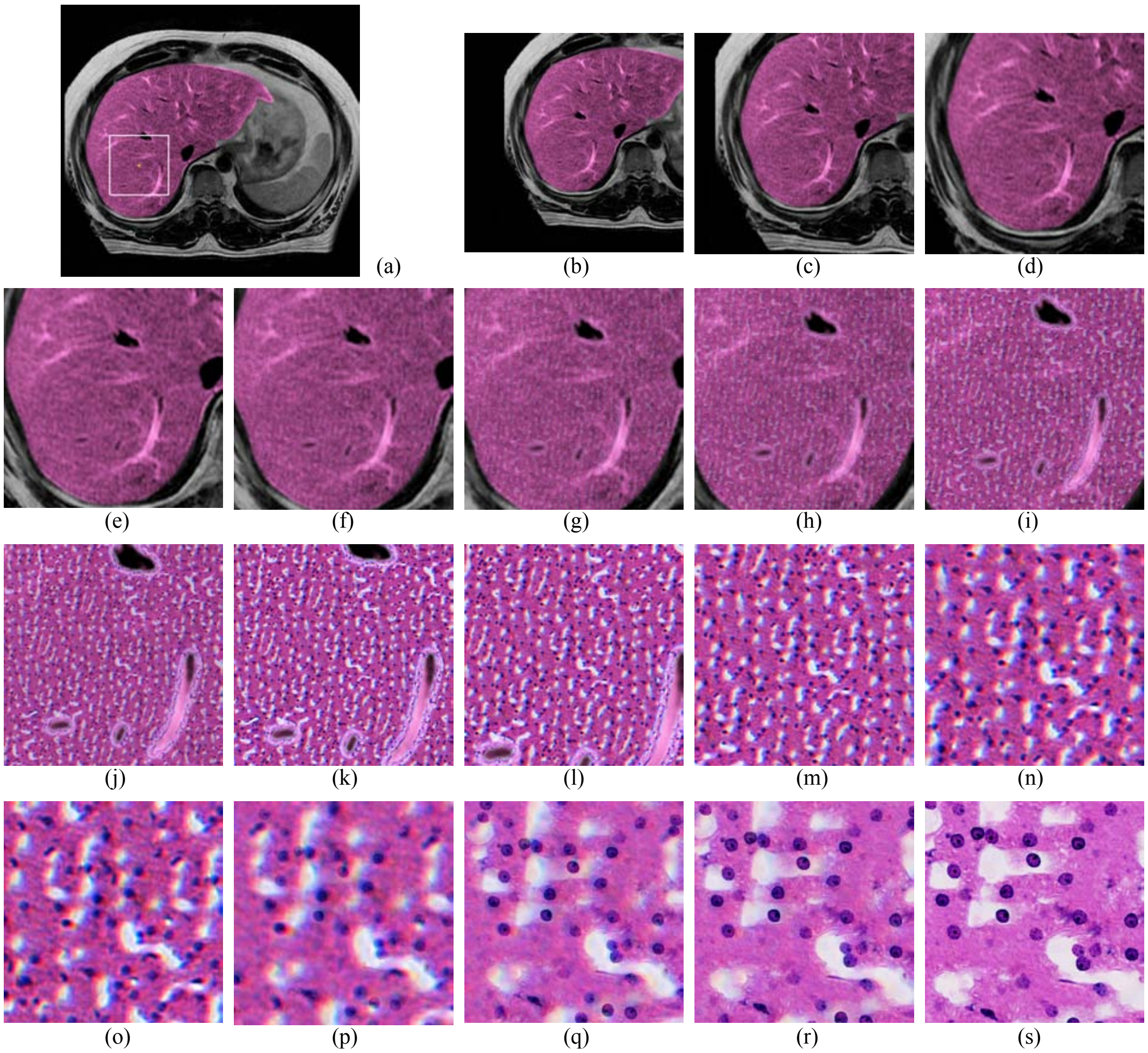


Figure 2: Illustration of the semantic zooming capabilities facilitated by the virtual microscope, using a human liver as an example:
(a) MRI image of a liver, where the white square is the user-specified region of interest,
(b)-(s) a typical image sequence during a semantic zoom, in which (k) is the synthesized histology level image, and (s) is the synthesized cell level image,
(c)-(e) magnified MRI level images, (l)-(p) magnified histology level images,
(f)-(k) images obtained by blending magnified MRI and minified histology level images,
(o)-(s) images obtained by blending magnified histology and minified cell level images.

however, one in which large-scale features and its small-scale detail smoothly blend into one another.

For example, one of the possible domain applications of our system is the "virtual microscope", where users start at a low-resolution MRI or CT image of some biological tissue and then slowly zoom in anywhere they desire to reveal the underlying cell structure, and finally the interior of the individual cells themselves. This process is illustrated in Fig. 2, for a human liver. Other possible applications include multi-resolution viewers for terrains, the universe, a sheet of metal, or any other domain that offers multiple levels of semantically constrained data, under the assumption that these data can be obtained. The fact that the different levels are obtained via synthesis and not via filtering of a common source imposes certain restrictions on the use of our technique. For example, our medical viewer would not be suitable for diagnosis of a diseased liver. However, it could be employed in a surgical simulation trainer, an electronic atlas for medical students, or a scientific illustration tool. Note that in these application scenarios the data at the different levels of scale do not have to be acquired from the same specific object, or in this case, person. This is especially advantageous since some of the higher resolution acquisition methods may be destructive. Similar restrictions and applications can also be envisioned for other application domains.

Our paper is structured as follows. First, in Section 2, we will elaborate on related work and then present our contribution. Section 3 will illustrate our 2D system by a specific example. The 3D extension is discussed in Section 4, and Section 5 will end with conclusions.

2 RELATED WORK

There has been much research focusing on texture synthesis approaches and applications in recent years. Texture synthesis algorithms take sample images as input and synthesize new images with similar textures. These algorithms can be roughly classified into three categories: statistical, pixel-based and patch-based texture synthesis. They have been found very helpful in image processing and also have some exciting extensions, such as surface texture synthesis [19][22], temporal textures synthesis [17][21][12], reflectance texture synthesis [18], and others. In the following, we only discuss the synthesis approaches and applications most related to our work.

Pixel-based synthesis algorithms synthesize textures pixel by pixel, which makes them rather flexible and easy to extend and apply to different areas. The representative algorithms include: Efros/Leung's non-parametric sampling algorithm [6], Wei/Levoy's multiresolution synthesis algorithm [21], which performs exhaustive search and accelerates based on tree structured vector quantization (TSVQ). Ashikhman's coherent synthesis [1] and Tong's k-coherent synthesis [18] algorithms reduce the search space significantly. Their synthesis process is faster, but only suits particular types of textures well. Hertzmann's Image Analogies algorithm [10] combines [21] and [1], uses PCA, and approximates nearest neighbor search (ANN) to accelerate the search process, offering better results. Zelinka and Garland [27] synthesize textures in real-time using a Jump Map, after a relatively slow analysis process. However, many pixel-based approaches suffer from image blurring and garbage growing.

Compared to pixel-based algorithms, patch-based synthesis algorithms tend to be faster and more stable, and do not suffer from blurring and garbage growing. They are, however, less flexible since they generate textures by copying whole patches from the input. Xu's chaos mosaic [23], Efros/Freeman's image quilting [5], Liang's [13], and Kwatra's Graphcut [12] algorithms all belong to this category. We use image quilting in our system, since it is efficient and also easy to implement. For 3D applications, Graphcuts [12] also seem to be a good choice.

For our application, we combine pixel-based synthesis [21], image quilting [5], and our pattern-based synthesis. Our approach is fundamentally different from that of Nealen and Alexa [15] who use pixel-based re-synthesis to eliminate remaining errors in the overlap regions of patch-based synthesis. In contrast, we apply different types of synthesis methods to synthesize different regions and features in an image. Further, our pattern-based synthesis is location constrained and differs from the algorithms based on pattern placement in the surface texture synthesis domain, such as pattern-based texturing revisited [16], and texture particles [3].

In our application, we make frequent use of *constrained texture synthesis*, where the patch selection and texture generation is made dependent on some underlying constraints. This technique has been utilized in image processing, such as image restoration [24] and texture transfer [1][2][5][10]. Another example is the texture-by-numbers technique [10], which is able to perform synthesis from images in which the texture distribution is not stationary but is based on the labeling of the component textures of images. These label images, representing the segmentation information of images, are created beforehand, possibly by the user. Some automatic color or texture segmentation methods are used for guiding the texture synthesis process in [11][4]. Our constrained texture synthesis follows a similar idea, but here only the segmentation of the sample images can be performed in advance. The features or patterns in the synthesized images have to be detected and labeled automatically when they are needed during zooms (see Section 3.2 for further detail). To enable proper semantic relationships across zoom levels, component textures should be placed carefully, following certain constraints including color, intensity, distance fields, location, and features/patterns of the image.

In contrast to Freeman's super-resolution algorithm [8] which generates enlarged images on the same semantic level than the base image, our application performs enlargement/zooming [14] that spans several semantic levels. Our main contributions are:

- *Semantic zooming* uses texture synthesis to extend image-guided detail enhancement to multiple levels of scales.
- *Constrained texture synthesis* facilitates smooth semantic evolution and detailing of features across zoom levels.
- *Feature-guided texture synthesis* considers the properties of features or patterns in the image at a certain semantic level and chooses image quilting, pixel-based, or pattern-based texture synthesis methods in accordance with the region's synthesis requirements.

3 THE VIRTUAL MICROSCOPE — A 2D VIEWER

We first discuss the 2D application, which acts like a microscope with a wide range of magnification. Then, in the next Section, we will discuss its extension to 3D. A system overview is shown in Fig. 3. First the underlying multi-resolution image data are collected and preprocessed to build a set of sample images. Then the sample images are analyzed to choose the appropriate texture synthesis approaches and constrained rules for each pair of adjacent levels. All these are stored in a small database, which will be used during the semantic zoom operation.

At the beginning, the user views the image at the coarsest resolution (Fig. 2a). Once the user specifies a region of interest in this image and zooms in, this part of the image is gradually magnified. When the image magnification reaches a certain scale, the image detail of the next level is generated through semantically constrained texture synthesis based on the currently magnified image region. For instance, when the user zooms into the image from the MRI level to the histology level, the system needs to synthesize the corresponding histology level image. The same is the case for the cell level. Blending of two consecutive levels enable the system to go smoothly from small-scale features to high-scale features. Thus, there are three main tasks in our system: data preprocessing,

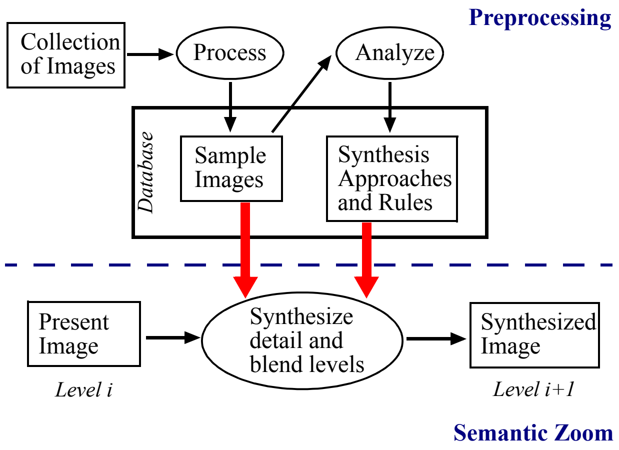


Figure 3: System Overview

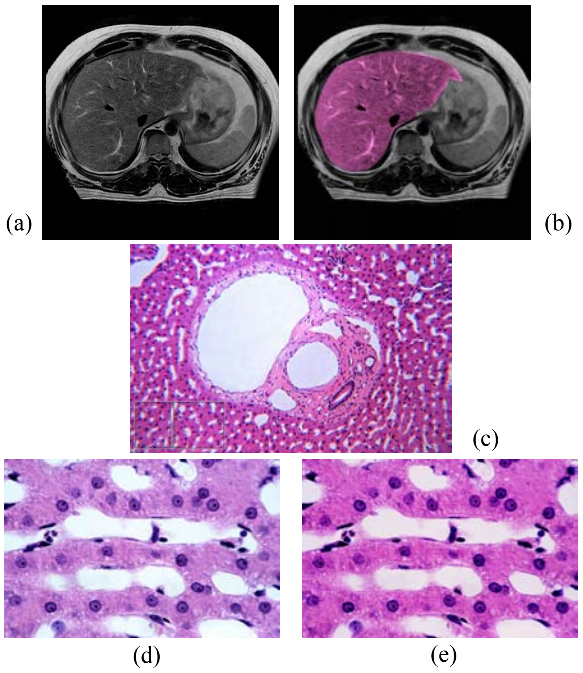

(a) (b) (c) (d) (e)

Figure 4: Image data and pieces of colorized sample images. (a) MRI liver image, (b) colorized, (c) low-scale histology image, (d) high-scale histology image, (e) colorized (Images (c) and (d) courtesy of http://www.bu.edu/histology)

constrained texture synthesis, and level blending. We will now describe each of them in detail.

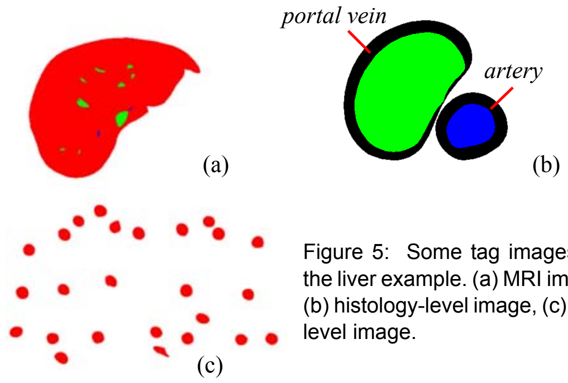
3.1 Preprocessing

We first need to collect data corresponding to the various levels and perform some amount of preprocessing on them. Fig. 4 shows the sample images used in the liver example: an MRI image, a low-scale histology image, and a large-scale histology image. These three levels will be referred to as MRI level, histology level, and cell level, respectively. However, it is easy to increase the number of levels as long as the corresponding texture data are available. Once the images have been collected the following pre-processing steps have to be performed.

Colorization: Typically, the images that are collected have different colors. In order to reduce the distinct discontinuities arising from mismatched colors during zooms, we need to match the colors across levels. The color correction can be easily done by image processing methods or tools, such as Adobe Photoshop. The colorized images shown in Fig. 4 are the sample images that will be used to guide the synthesis later on. Since we use the color of the low-scale histology image for transfer, this image requires no change.

Segmentation: The sample images need to be segmented into prominent features or patterns, based on color, shape, or pre-knowledge. In our particular example, for the MRI image, we segment out the liver region as well as the portal vein and the artery elements. The segmentation can mostly be done via image processing methods [9] or tools. The segmentation results, which will later help us to match texture synthesis methods with different features or patterns, are stored in tag images (see Fig. 5).

The data preprocessing is the only part in our system which may require some manual work to refine the image processing results, but it needs only to be done once. After that, no manual



(a) (b) (c)

Figure 5: Some tag images for the liver example. (a) MRI image, (b) histology-level image, (c) cell-level image.

work is required. The colorized sample images and the corresponding tag images are then stored in a database.

3.2 Constrained Texture Synthesis

3.2.1 Synthesis approaches

As mentioned before, a variety of texture synthesis approaches could be applied to generate the image detail for semantically different levels. For each pair of adjacent levels, which texture synthesis approaches should be used depends on the texture features, and the region in which the texture will grow.

- If the texture is isotropic, semi-structured, or structured, and grows in a large region, image quilting or other patch-based algorithms produce better quality results than pixel-based methods. The primary parameters in image quilting include patch size and overlapping region size. Both mainly depend on the prominent structures of the texture and should be decided before synthesis.
- If the texture has layers and/or grows within a small irregularly shaped region, then a modified pixel-based approach forms a convenient way to add fine detail in the magnified images. We give the details of our algorithm in Section 3.2.3. The parameters in a pixel-based synthesis algorithm [21] include the shape and size of the pixel neighborhood, as well as the number of levels if a multi-resolution algorithm is applied.
- If the texture is composed of atomic patterns which should be preserved during synthesis, our pattern-based synthesis is employed to synthesize the patterns, while other pixel-based or patch-based approaches can be applied to synthesize the background color (see Section 3.2.4 for further details).

Why do we need *constrained* texture synthesis? We need it to ensure that the generated textures on one level are semantically consistent with the level before. Since we use level blending to facilitate intermediate zooms, this is obviously very important. Standard texture synthesis algorithms only use the present layer information in the generation process, and Fig. 6 demonstrates the poor blending that will occur if we perform texture synthesis on the histology level without constraining it to the lower-scale MRI level. Similar problems arise for the cell level and the lower-scale histology level. Thus, textures of the high-scale image should always be synthesized to match the features of the low-scale image under specific constraints. For this reason, the system always computes a tag image of the current result image to facilitate the matching process. This is somewhat similar to the label-constraints used in [10] and [1], but in our application the constraint tags are not specified by the user but generated automatically, using image processing techniques.

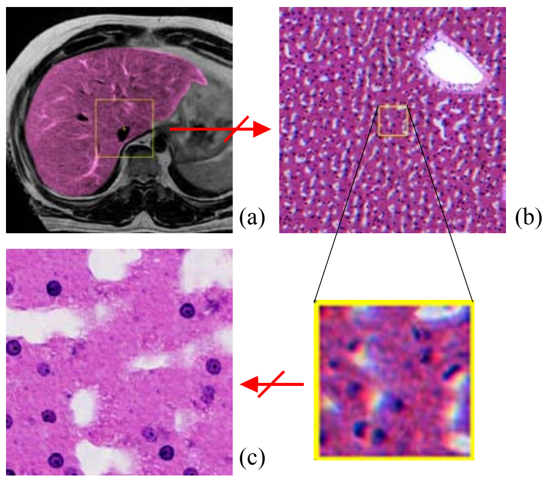


Figure 6: Mismatched levels. The histology level image (b) does not match the specified region of the MRI level image (a), and the cell level image (c) does not match the specified region of (b) either.

In our system, three texture synthesis methods are combined to synthesize the image. We mainly discuss the algorithm modifications which need to perform constrained synthesis.

3.2.2 Constrained image quilting

Image quilting is used to generate the background texture for the histology level and the cell level image, but other patch-based synthesis methods, such as Graphcuts, may also work. In the histology level, background is defined as everything except the vessels and their surrounding layer. In the cell level, background is defined as everything except the cells, the vessels and their surrounding layer. We also tried pixel-based synthesis methods to generate the background as well, but neither the single resolution nor the multiresolution (with TSVQ acceleration or PCA and ANN acceleration) algorithm seemed to work well for the textures used here, mainly because the features in the texture tended to come out blurred.

Our constrained quilting algorithm differs from typical quilting in the following two ways. First, not all patches in the segmented sample image can be used for synthesis. For example, at the histology level, the textures around the portal vein and the artery are different from the background texture (see Fig. 4c). Hence, the patches falling into those regions should not be used to generate the background texture. Second, both patch placement and selection are constrained to satisfy the match requirement. Especially at the cell level, in order to match the histology level features, the quilting process is constrained by the color/intensity of the magnified histology level image. An example for this are the white areas, called *sinusoids*, which appear on both synthesized levels and should be matched. Thus, when selecting a candidate patch for the third level, the location, shape, and distribution of its sinusoids must match that of the corresponding second-level area. This is not a limitation since our sample database is diverse enough, and we have never encountered a case where no fit could be found. Considering the texture structure size, the quilt patch size is chosen to be 40×40 pixels, and the overlapping width is 6-8 pixels.

A further constraint for background texture synthesis are object boundaries, both interior and exterior. The tag images play an important role in complying to these boundary constraints, and this will be discussed at the end of this section.

3.2.3 Constrained pixel-based synthesis

Smaller structures constrained to tight and curved boundaries are better generated using pixel-based synthesis methods, since patch-based methods work on a scale too large to adhere well to the object's geometry. In our application, we use this type of approach to generate the small textures in the surrounding layer around the portal veins. However, at the same time it is desirable to transfer the global characteristics of the sample texture to the output image as well. For example, texture features, such as smooth muscle cells in our application, which are closer to the object boundary in the sample should also be placed closer to the boundary in the output image. We can achieve this by constraining the texture generation process by a measure imposed by the object geometry – distance fields, which we use here to (i) constrain the texture generation and (ii) help to find the outside boundaries for magnified veins to guide the synthesis process. We will first illustrate our pixel-based algorithm for the general case (see Fig. 7) and then discuss how it is applied within a specific example.

We calculate the distance field using a $\sqrt{2}$ distance transform and normalize it to a range of [0,1]. The distance field is shown in Fig. 7a as a grey image, in which pixel value maps to distance. If the given sample texture has a layered appearance (Fig. 7a), then the synthesis process must depend on these distance values. After calculating the distance fields for both sample and result image, we use the standard scan-line order to synthesize the pixels. There, for

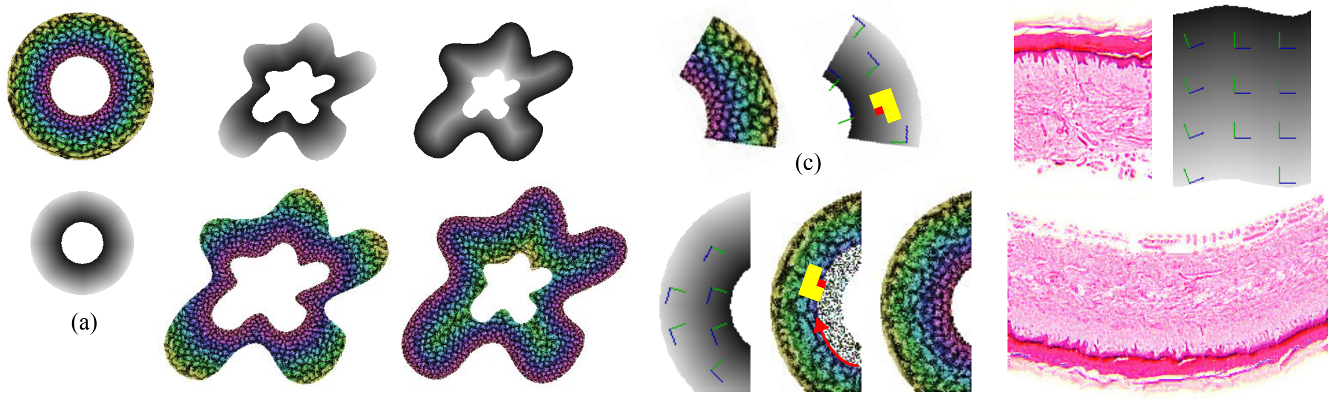


Figure 7: Our pixel-based synthesis methods. Pixel synthesis based on distance field: (a) sample image and its distance field, (b) reference distance fields and corresponding synthesis results. Pixel synthesis based on distance field and gradient field: (c) sample image and its distance and gradient fields, (d) synthesis process and result, (e) sample image and the result of synthesizing a thick skin histology image.

each pixel in the result image, the matched pixel must be chosen from the set of pixels that (i) observe the usual texture synthesis metrics [21] and (ii) have a similar distance field value.

If the input image is part of a layered texture, or if we want to reduce the sample image size to speed-up the synthesis, our pixel-based synthesis method will not only depend on the distance value, but also on the texture direction, which is calculated from the distance field and represented by a gradient field (see Fig. 7c). The pixel synthesis order depends on the distance values, and, based on the gradient, rotated L-neighborhoods are compared to find the best match.

In our bio-tissue example, we pre-compute the normalized distance field around the portal vein based on the tag image of the sample histology level image (see Fig. 8a). When synthesizing the histology level image, we compute a similar distance field around the vein of the magnified MRI image to find the boundary of the vein structures (Fig. 8b). The detail in the vein periphery is then synthesized based on the distance and gradient values.

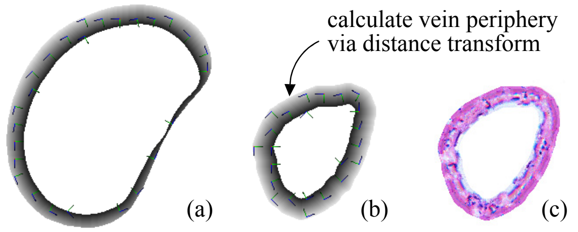


Figure 8: Vein periphery synthesis based on distance fields. (a) generated from the segmented sample image, (b) generated from magnified MRI image, (c) texture detail.

3.2.4 Pattern-based synthesis

Our pattern-based algorithm is designed to preserve potential atomic structures, i.e. structures that cannot be cut, such as cells. Pixel-based or patch-based synthesis methods cannot generally guarantee that features remain uncut or undistorted, since they have no knowledge about which part of the texture constitutes a whole atomic pattern. We require an algorithm that will ensure that atomic structures remain intact and, at the same time, satisfy the match requirements.

We can achieve this by identifying the location of the atomic structures on the low-resolution level and replace them by high-resolution versions in the magnified level. If these structures have fuzzy boundaries that blend with the background, it is useful to keep these as well. They can then later help to integrate the features into the background in a coherent way.

The first step involves identifying the atomic features. In our liver tissue example, these atomic features are represented by the cells in the cell level sample image (Fig. 4e) and are segmented as patterns (Fig. 5c). When synthesizing the cell level image, the algorithm first detects all possible cells (dark points) of the magnified histology level image based on the image intensity, and records this location information. We detect the dark points using two thresholds. Then location-constrained pattern placement proceeds, and the cell patterns are chosen randomly to increase the variation of the result. A similar method can also be used for magnifying the cells in the layer around the portal veins.

As we have mentioned above, the tag image, which corresponds to the current zoomed image, is important to comply with the match requirement. For example, the vessels (portal veins and arteries) represent interior objects which should be preserved as they are and properly scaled under zoom. However, scaling the tag image presents a problem. When the image is magnified, the corresponding tag image should also be enlarged at the same rate. Without any specific process, the boundary of the enlarged tag image will have a binarized effect (Fig. 9a). To prevent this, we use a smooth interpolator for the tag values, and then choose an intermediate value as the threshold to decide the boundary. Using this procedure, the magnified image will still have a smooth boundary (Fig. 9b). Another possible solution is to represent the boundary as a spline curve. If the segmentation information is stored using a spline curve, the enlarged spline curve can be calculated based on several control points while the image is magnified. In this way, the boundary can be very accurate.

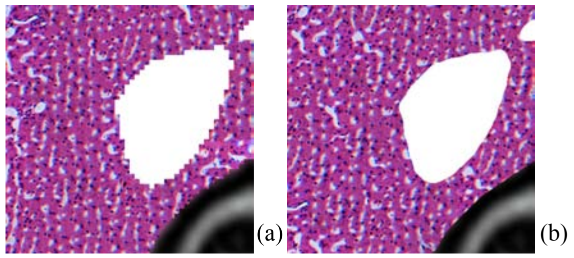


Figure 9: Smooth boundary problem caused by tag image magnification: (a) dentate boundary, (b) smooth boundary.

3.3 Smooth Semantic Zooms

When zooming into a specific region of the image, our system combines two processes: (i) magnification of the current level image, and (ii) minification of the synthesized next high-scale level image. This achieves any level of magnification from only a few images with different semantic detail.

The system has a number of parameters, some are set by the user and some are decided by the available data. The first such parameter is the size of the output image, $M \times M$, which specifies the screen size of the microscope. A second parameter is the maximum zoom scale Z_{max} for each level, which is determined by the resolution of the subsequent, more fine-scale level. This factor determines the amount of standard magnification that needs to be performed using the current level data before new semantic detail can be filled in by synthesizing from next-level data. Obviously, the more levels are available, the less blur will be encountered when zooming in. Since for real optical, confocal, or electron microscopes the maximal zoom scale can be from thousands to millions, our application accelerates the zooming activity by dramatically reducing Z_{max}. When the present level data is magnified at Z_{max}, the resolution has been reached at which the next higher level data can be synthesized to provide the missing detail.

Also, at the beginning, the user specifies a zoom focal point *F*, which determines the center of the region of interest *R*. This region *R* has a size $N \times N$ and is calculated by the system, such that $N = M/Z_{max}$. *R* marks the image region that will be replaced by the next higher level detail when the zoom scale *Z* reaches Z_{max} (in our example, this region is shown as the white square in Fig. 2a).

The last parameter that our system maintains is the view port *VP* which is centered at *F* and has a size $V \times V$. It varies with *Z*, such that $V = M/Z$. At any given *Z*, the system will capture the image inside the *VP*, and then magnify and fit it into the output image. At startup, the image is not magnified, i.e. $V = M$ and *Z*=1, and is shown as the output image directly (Fig. 2b). When the image is gradually magnified by the user, *Z* increases, while the *VP* decreases. Once the *VP* has reached *R*, synthesized image data due to the next higher-level detail should be made available.

It is desirable to avoid a sudden change of the display, where the image generated from the next higher level of resolution suddenly pops in. We accomplish a graceful transition by blending the images of two consecutive levels over some range of zooms, properly weighted by a zoom-related weighting function. In addition, we prefer to do this without having to view blurred features of the present level. We can achieve both of these requirements by specifying a transition point *t* with a zoom scale Z_t, where $Z_t < Z_{max}$, at which we compute the image for the next level, minify it, and blend it with the magnified present level. This early computation of the high-resolution image, however, requires the computation of extra data at boundaries, later culled with further zooming until the $Z = Z_{max}$. More specifically, suppose that the synthesized image has size $S \times S$, then $S = M \times Z_{max}/Z_t$. The advantage of having a larger image available is that it allows more panning activity within the next semantic level.

The smooth image transition process over a range of consecutive zooms is illustrated in Fig. 10 below. After the transition point, the magnified present image and minified synthesized image are smoothly blended by gradually changing their weights inversely, i.e. the magnified image will fade out while the synthesized image will fade in.

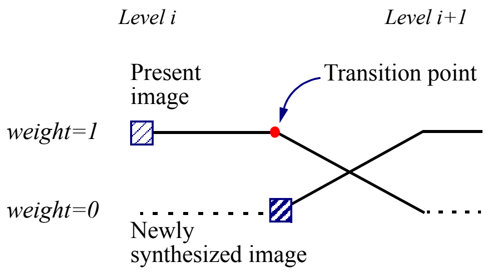


Figure 10: Image transition process.

3.4 Results

In this section we report on our specific application – the virtual microscope viewing a liver datasets at three levels of semantic scales. The sample images and corresponding tag images stored in our database have been shown in Fig. 4 (b,c,e) and Fig. 5. A few frames of the resulting image sequence during a semantic zoom are shown in Fig. 2. When the user specifies a region-of-interest in the MRI image of a liver and zooms in, then this part of the MRI level image is gradually magnified and blended with the synthesized histology level image. If the user further zooms in from the histology level, the histology level image is magnified and eventually blends with the synthesized cell level image. This resembles the functionality obtained with a real microscope, when slowly examining an interesting part of a liver. Besides zooming, the user can also pan to inspect nearby regions.

In our algorithm, once the sample images are chosen, the time to synthesize a certain level image mainly depends on the output image size *M* and the magnification scale Z_t of the transition point. When *M* is fixed, the time spent on synthesis and the blending process can be adjusted by Z_t. For example, suppose the output image size is fixed on 400×400 and the maximal scale Z_{max}= 4. If the specified scale Z_t of the transition point is 2, then the synthesized image has a size of 800×800. With the current implementation, it will take several minutes to generate the result image. If Z_t is increased to 3, the corresponding synthesized image becomes 533×533, which reduces the time spent on synthesis. However, the blending effect is also reduced, which means the synthesized next-level image will pop in more abruptly.

4 EXTENSION TO 3D

The idea extends well to volumetric data. In order to generate subresolution detail for volume data, we extend image quilting to volume quilting, and also apply a 3D pixel-based synthesis algorithm. In volume quilting, we apply the graph cuts algorithm [25][26] to find the best seam surface between two neighboring blocks, instead of using the shortest path algorithm, which is applied in image quilting but not easy to be extended to 3D [12].

From the Visible Man's cryosection data, we reconstructed the volume and segmented out the liver. Similar to the 2D case, the volume data is also colorized to match the histology data. The sample histology volume is built based on the features in the 2D image and certain 3D growth rules. We could also apply Wei's solid texture synthesis method [22] to generate a sample volume, however, it is difficult to get a high quality solid texture. Fig. 11 shows the volume data required by the synthesis procedure.

In the 3D extension of our viewer, the user specifies a volume-region-of-interest (Fig. 12a), and this volume region is cut out from the original volume and rendered. During 3D zooms, the volume region is magnified and smoothly blended with the minified synthesized higher level volume. The observed volume size changes during zooms, in contrast to the fixed-size output images in the 2D system. Some volumetric semantic zooms are shown in Fig. 12. For the histology level, as in 2D, the textures around the vein are synthesized by a pixel-based algorithm, while other textures are created by volume quilting. Fig. 13 shows volume with cut and translucent rendering results. The translucent volumes are rendered using the OpenQVis software (http://openqvis.sourceforge.net/). An advantage of volume synthesis over traditional surface synthesis is that only the former can illustrate the translucent effect of internal structures.

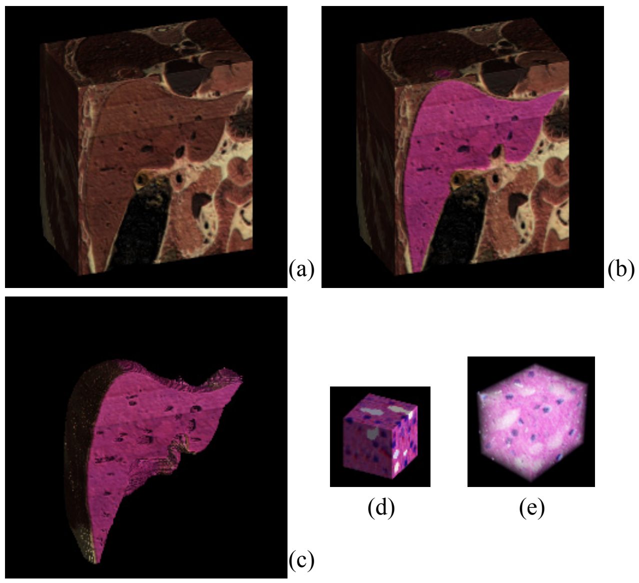


Figure 11: Volume data and colorized volume. (a) visible man's volume, (b) colorized volume, (c) segmented liver, (d) an example of the sample histology level volume and its translucent result (e).

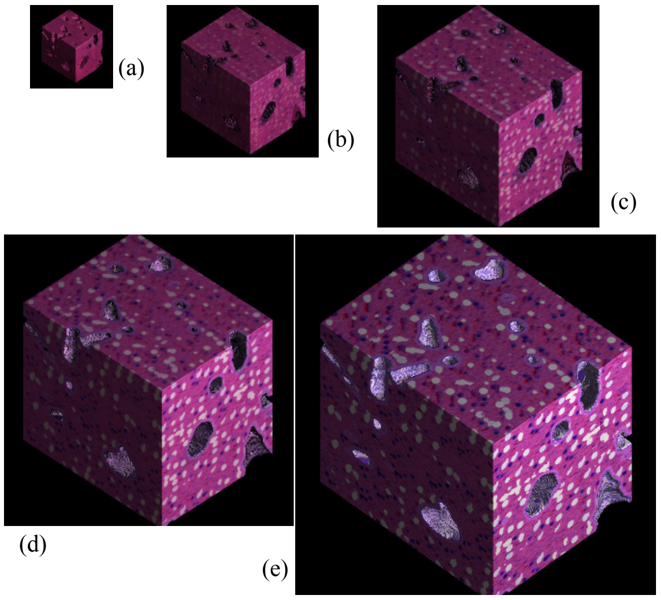


Figure 12: Illustration of semantic zooming into volume data. (a) first level for part of the liver, (e) histology level of (a), (b)-(d) volumes obtained by blending the magnified first level volume and minified histology level volume.

5 CONCLUSIONS

We have described a new constrained multi-scale texture synthesis method to facilitate semantic zooms. Pixel-based, image quilting, and pattern-based synthesis methods were unified to generate high-detail images under certain constraints. Our demo application, a virtual microscope, demonstrated that quite interesting and useful image sequences can be generated using our framework.

In future work, we would like to improve our algorithm in terms of accuracy and speed. For the former, more sophisticated segmentation and constraints may yield more refined small detail. We would also like to explore better interpolation methods for the oriented texture synthesis to overcome some of the remaining visual artifacts. Finally, optimization and GPU acceleration of our algorithm will provide more interactive capabilities, i.e. for generating the detail on demand when zooming into an image or volume.

ACKNOWLEDGEMENTS

Support was provided by NSF Career grant ACI-0093157. We would like to thank Dimitris Samaras and Neophytos Neophytou for their help in the preparation of an early version of this paper and the anonymous reviewers for their thoughtful comments.

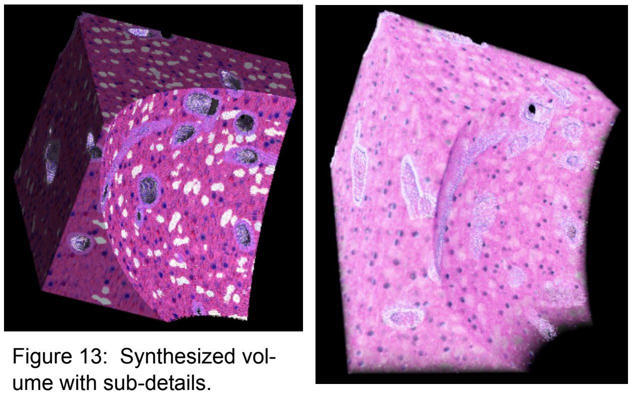


Figure 13: Synthesized volume with sub-details.

REFERENCES

[1] Ashikhmin, M. 2001. Synthesizing natural textures. *ACM Symposium on Interactive 3D Graphics*, 217-226.

[2] Ashikhmin, M. 2003. Fast texture transfer. *IEEE Computer Graphics and Applications 23, 4*, 38-43.

[3] Dischler, J. M., Maritaud, K., Ievy, B., and Ghazanfarpour, D. 2002. Texture particles. *Computer Graphics Forum 21, 3*, 401-410.

[4] Drori, I., Cohen-or, D., and Yeshurun, H. 2003. Fragment-based image completion. *Proc. of SIGGRAPH 2003*, 303-312.

[5] Efros, A. A., and Freeman, W. T. 2001. Image quilting for texture synthesis and transfer. *Proc. of SIGGRAPH 2001*, 341-346.

[6] Efros, A. A., and Leung, T. K. 1999. Texture synthesis by non-parametric sampling. *Int. Conference on Computer Vision, 2*, 1033-1038.

[7] Furnas, G. W., and Bederson, B. B. 1995. Space-scale diagrams: understanding multiscale interfaces. *Proc. of CHI'95 Human Factors in Computing Systems*, 234-241.

[8] Freeman, W. T., Jones, T. R., and Pasztor, E. 2002. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 56-65.

[9] Gonzalez, R. C., and Woods, R. E. 1992. Digital Image Processing. Addison-Wesley Pub Co., Boston, MA.

[10] Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B. H., and Salesin, D. 2001. Image analogies. *Proc. of SIGGRAPH 2001*, 327-340.

[11] Jia, J., and Tang, C.-K. 2004. Inference of segmented color and texture description by tensor voting. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 26, 6*, 771-786.

[12] Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *Proc. of SIGGRAPH 2003,* 277-286.

[13] Liang, L., Liu, C., Xu, Y. Q., Guo, B., and Shum, H. Y. 2001. Real-time texture synthesis by patch-based sampling. *ACM Trans. on Graphics 20, 3*, 127-150.

[14] Perlin, K., and Fox, D. 1993. Pad: an alternative approach to the computer interface. *Proc. of SIGGRAPH 1993*, 57-64.

[15] Nealen, A., and Alexa, M. 2003. Hybrid texture synthesis. *Proc. Eurographics workshop on Rendering 2003,* 97-105.

[16] Neyret, F., and Cani, M. P. 1999. Pattern-based texturing revisited. *Proc. of SIGGRAPH 1999*, 235-242.

[17] Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. 2000. Video textures. *Proc. of SIGGRAPH 2000*, 489-498.

[18] Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., and Shum, H. Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. on Graphics (SIGGRAPH 02)*, *21, 3,* 665-672.

[19] Turk G. 2001. Texture synthesis on surfaces. *Proc. of SIGGRAPH 2001*, 347-354.

[20] Wolberg, G. 1990. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA.

[21] Wei, L. Y., and Levoy, M. 2000. Fast texture synthesis using tree-structured vector quantization. *Proc. of SIGGRAPH 2000*, 479-488.

[22] Wei, L. Y. 2001. Texture synthesis by fixed neighborhood searching. Ph.D. Thesis. Stanford University.

[23] Xu, Y. Q., Guo, B., and Shum, H. Y. 2000. Chaos mosaic: fast and memory efficient texture synthesis. *Technical Report MSR-TR-2000-32,* Microsoft Research.

[24] Yamauchi, H., Haber, J., and Seidel, H.-P. 2003. Image restoration using mutli-resolution image synthesis and image inpainting. *Proc. Computer Graphics International 2003*, 120-125.

[25] Yuri, B., and Kolmogorov, V. 2001. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 359-374.

[26] Yuri, B., Veksler, O., and Zabih, R. 1999. Fast approximate energy minimization via graph cuts. *International Conference on Computer Vision, 1*, 377-384.

[27] Zelinka, S., and Garland, M. 2002. Towards real-time texture synthesis with the jump map. *Proc. 2002 Eurographics Workshop on Rendering Techniques*, 99-104.