

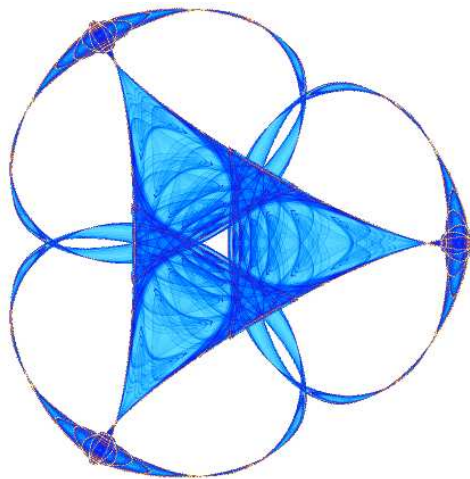
SUPER-RESOLUTION TEXTURING FOR ONLINE VIRTUAL GLOBES

By

Diego Rother
Lance Williams
and
Guillermo Sapiro

IMA Preprint Series # 2208

(May 2008)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA
400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455-0436

Phone: 612/624-6066 Fax: 612/626-7370

URL: <http://www.ima.umn.edu>

Super-Resolution Texturing for Online Virtual Globes

Diego Rother
University of Minnesota
diroth@umn.edu

Lance Williams
Google, Inc.
lancew@google.com

Guillermo Sapiro
University of Minnesota
guille@umn.edu

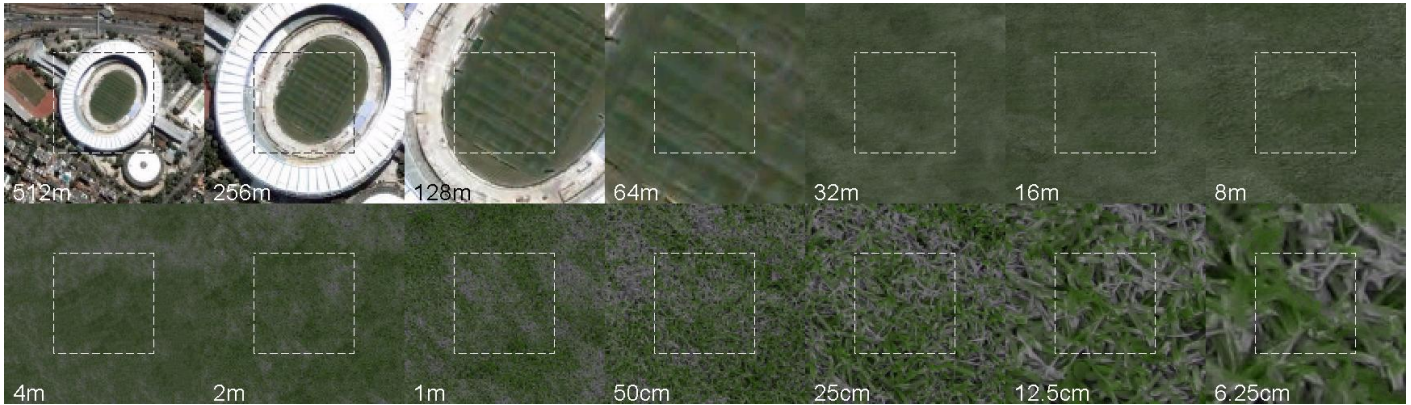


Figure 1: “Powers of two.” This sequence of images illustrates the framework proposed in this article to supplement and enhance the imagery of virtual globe applications (e.g., *Google Earth*). The first four images were extracted from *Google Earth*, the rest were synthesized with the proposed framework following the user’s zoom-in request. See the corresponding video at [28].

Abstract

Online virtual globe applications such as *Google Earth* and *Maps*, *Microsoft Virtual Earth*, and *Yahoo! Maps*, allow users to explore realistic models of the Earth. To provide the ground-level detail of interest to users, it is necessary to serve and render high resolution images. For planetary coverage at high resolution, a very large number of images need to be acquired, stored, and transmitted, with consequent high costs and difficulty for the application provider, often resulting in lower than expected performance. In this work we propose a supplementary approach to render appropriate visual information in these applications. Using super-resolution techniques based on the combination and extension of known texture transfer and synthesis algorithms, we develop a system to efficiently synthesize fine detail consistent with the textures served. This approach dramatically reduces the operational cost of virtual globe displays, which are among the most image-intensive applications on the Internet, while at the same time improving their appearance. The proposed framework is fast and preserves the coherence between corresponding images at different resolutions, allowing consistent and responsive interactive zooming and panning operations. The framework is capable of adapting a library of multiscale textures to pre-segmented regions in the highest-resolution texture maps available. We also describe a simple interface to obtain class label information from contributing users. The presentation of the constituent techniques is complemented with examples simulating our framework embedded in *Google Earth*.

1. Introduction

Several online virtual globe applications currently allow users to explore realistic models of the Earth [1]. Examples of these applications include *Google Earth*TM [2] and *Google*

*Maps*TM [3], *Microsoft Virtual Earth*TM [4], *Yahoo! Maps* [5], *Earth Explorer* [6] and *World Wind* [7]. These applications are supported by some of the largest organized collections of imagery on the Internet.

Online virtual globes commonly consist of a *client* application, which is a special program or simply a web browser running on the user’s machine, and a *server* application, running on the provider’s machine. The client receives navigation commands from the user and creates the corresponding display. If the client does not have the information required to create the display, it issues a request for this information to the server. To answer the client’s requests, the server imagery is organized as a clipmap pyramid [8], containing the lowest resolution image of the Earth in the highest level, an image of double that resolution in the next level, and so on, up to the highest resolution image served.

As the user zooms in, increasingly higher resolution images (stored in lower levels of the pyramid) need to be accessed and transmitted to the client. When the stored resolution is exhausted, the highest-resolution pixels are simply interpolated, producing vague, blurry vistas, often with a visible grid structure due to Mach-banding between samples. Texture-mapped building models may be instanced on the landscape, but elsewhere, high spatial frequencies useful for interactive navigation simply vanish. The interpolated imagery does not have the statistical properties or visual characteristics of natural imagery; it typically resembles nothing in nature.

On the other hand, much of the surface of Earth exhibits rapidly changing and/or stereotypical texture, the specific details of which are not of general interest. Many areas such as desert sands or snow cover are capable of rapid change, rendering an expensive high resolution acquisition quickly outdated; while some areas, such as grasslands or croplands, have highly predictable or repetitive texture whose exact high definition details are irrelevant to the average user, in spite of the relevance of its identity (i.e., distinguishing a cornfield

from a soybean field may be relevant, observing the exact arrangement of leaves may not). A consistent depiction that portrays the correct region classes offers the user both satisfying visual detail, necessary for navigating the surface, and useful symbolic information. Global *literal* detail over some range of scales will never be available.

The fact that the surface covered by these provisional/predictable areas is by no means negligible, emphasizes the importance of augmenting a deep-scan approach. Seventy one percent of the planet’s surface is covered by water [9]. The remaining 29% is covered as follows: 32% forests and woodland, 26% permanent pastures [10], and 13.3% arable land (4.7% with permanent crops) [9]. By comparison, the size of urban areas (where high-resolution scanning generates more useful data) is just 1.5%. In the U.S. just four crops (corn, wheat, cotton and soybean) cover 10% of the country’s surface [11].

Since online virtual globe applications must be interactive, the proposed solution has to be fast, on the order of the time required to download equivalent images from the server, or faster. In addition, to create the feeling of seamless transition between layers in the pyramid (zooming in or out), the coherence between consecutive pyramid levels must be preserved, each layer integrating to the layer immediately above.¹

These circumstances lead us to propose in this work a system for continuation, on the client side, of texture details in images for the lower pyramid levels. We approach this using super-resolution image processing methods based on the extension of available texture synthesis and transfer algorithms. The proposed texture transfer component of the framework consists of several passes over the image. In the first pass, a modified Wei-Levoy texture synthesis algorithm, [12], primes the data for subsequent passes by linking pixels in the image to similar pixels in the “training” texture. Next, several passes based on modifications on Ashikhmin’s approach, [13], add structure in order to improve the appearance match to the training image texture. This combination avoids the relatively high cost and smoothed results of conventional Wei-Levoy iterations, while improving coherence between input and output images. In addition to these extensions and combinations of known techniques, we propose a novel rule to directly propagate texture down the pyramid in areas where inter or intra layer coherence conflicts cannot arise, further improving both the quality of the results and the computational speed.

The proposed framework dramatically reduces the storage space required for the clipmap pyramid, the Earth surface area to be acquired, the acquisition resolution, and the bandwidth required for transmission of these images to the client, while at the same time improving the appearance and information content of the models.

To apply super-resolution techniques and display to the user a reliable representation, the particular class label (e.g., “water,” “grass,” “asphalt,” etc.) of each pixel in the images must be known. We propose two strategies to obtain these

¹ Currently, online virtual globe pyramids may contain layers supplied by different providers or taken during different seasons, therefore the coherence between layers may not be preserved at all.

labels: 1) ask contributing users² to provide these labels; and 2) obtain them from agricultural mapping databases (e.g. [11]), which may in turn obtain this information from ground level surveys or automatic segmentation and classification of hyperspectral images. In this article we demonstrate the first strategy through a simple interface to obtain this information from users.

In many undertakings, users are enthusiastically providing the information delivered by the system, Wikipedia serving as a defining example. Closer to our current interest is Wikimapia [14], a website where users are encouraged to mark places and roads to “describe the whole planet Earth,” or Google SketchUp [15], in which users create 3D models that are later integrated into Google Earth. In our proposed system, contributing users are equipped to provide three pieces of information. First, they can assign class labels for regions of pixels in the highest resolution image available. To input the labels in our implementation, we use state-of-the-art interactive segmentation techniques [16].

Second, for each class label (or material) defined, the user provides one or more sample “training” images specifying the appearance of the material at chosen scales. We call these images *keyframes*, in analogy to the keyframes of traditional animation. The supplied images, K_1, \dots, K_n , define the texture’s appearance at discrete levels of resolution or *keyscales*, S_1, \dots, S_n . These images are stored in a *texture pyramid* (distinct from the pyramid storing the Earth images described above). Training images can be easily acquired at ground level with a standard camera (all the ground-level textures in this article were acquired by one of the authors, with an inexpensive conventional camera or downloaded from the Internet).

Third, for each keyframe texture, the user provides the approximate true world size represented by a pixel in the corresponding training image (e.g., in meters/pixel). Each size is used to place the corresponding training image in the appropriate level (keyscale) of the texture pyramid. We call this framework *texture keyframing*. The system then uses this information, through the super-resolution specification here introduced, to synthesize new levels in the texture pyramid. The last two pieces of information can of course be omitted, if the virtual globe application provider keeps a library of previously keyframed textures that the user can select from.

The remainder of this paper is organized as follows. In Section 2 we review in detail the relevant prior texture synthesis/transfer techniques that our framework relies on. In Section 3 we describe the proposed system and the modifications that we developed to adapt and combine these fundamental techniques. We include results in Section 4 and conclude with a discussion in Section 5.

2. Previous related work

Super-resolution³ is an image processing operation to

² It might be useful to distinguish between two different kinds of users interacting with the system. Normal users (or just “users”) are those that only obtain information *from* the system. “Contributing users,” on the other hand, also provide information *to* the system, and may require qualification.

³ This is sometimes referred to as single-frame super-resolution, to avoid confusion with other techniques which aggregate information across frames to increase the resolution of frames in a sequence.

synthetically augment the resolution of an image by supplementing its low frequency components with “corresponding” high frequency components. Several techniques have been proposed to address this operation, see [17] for a review. We are particularly interested in a general class of successful methods we will call *pixel substitution by context matching* (PSCM).

PSCM techniques receive in general an input image that has to be modified, and a set of “training” (or “example”) images specifying some characteristic (or statistic) of the desired result. The input image is then modified, one pixel at a time, in raster scan order. For each pixel p in the input image, a pixel q in one training image is selected such that their corresponding neighborhoods (or “contexts”) are “similar,” and the color/attribute at q is used to overwrite p . The modified input image (the “output” image) is returned as the result.

Important algorithms of this class were introduced by Efros and Leung [18] and Wei and Levoy [12], with the goal of texture synthesis. In the following, these algorithms will be referred to as Efros-Leung and Wei-Levoy, respectively. The Efros-Leung algorithm receives an empty image of the desired size and an example of the desired texture appearance, and fills the image with pixels sampled from the example. To obtain the value to place in the current pixel p , all previously synthesized pixels in a square window around p (“causal context” in the following) are used as the context of p , $C(p)$. This context is then compared against contexts similarly extracted from the training image, and the pixel q that has the closest context $C(q)$ to $C(p)$ is used to fill p . Specifically, in a straightforward generalization, q is selected as $\arg \min_q \|f(C(p)) - f(C(q))\|_2$, where $f(\cdot)$ is a transformation that maps the context to a different space containing only its relevant (from a perceptual point of view) information. Efros-Leung selected the transformation to be a Gaussian weighting of the pixels in the context, so that center pixels have higher influence on the distance.

The Wei-Levoy algorithm follows the same course as Efros-Leung, with two important exceptions: 1) the synthesis proceeds in a multiresolution fashion, and 2) it uses tree-structured vector quantization to speed up the search for the best context in the training texture. Multiresolution synthesis starts by synthesizing a low-resolution version of the texture, as in Efros-Leung. The following level, which has double the resolution of the previous one, is synthesized next, using as context the causal context at the current resolution, concatenated with the whole (corresponding) context already synthesized at the previous resolution. With this approach, smaller contexts can be used, often leading to a faster implementation.

Ashikhmin, [13], suggested a simple but important modification to these techniques, further accelerating the synthesis, while at the same time improving the results for the important class of natural (or even irregular) textures. He noted that a potentially good candidate, q , to fill the current pixel p , can be obtained from the candidate q' in the training image, used to fill in p 's neighbor p' , appropriately shifted by $p - p'$ ($q = q' + (p - p')$), see Figure 4 in [13]). Since p' can be any neighbor of p , all and only the neighbors of p are used

to suggest candidates, avoiding the expensive step of searching the training image for an appropriate location to copy from. To support this, a matrix Q storing the original location in the training image of each copied pixel has to be kept ($q = Q(p)$). This “cloning” variation will be referred to as “Ashikhmin” in the following. This approach produces irregular patches⁴ that are copied from the training image to the resulting texture, avoiding the undesirable smoothing sometimes found in Efros-Leung or Wei-Levoy’s results.

The candidate locations to copy from are set at the start when the matrix Q is initialized. Multiple iterations of this algorithm can be performed. Individual patches can grow or shrink, or even disappear, at each iteration, but no patches can be spontaneously created; whether a patch is included is defined when the matrix Q is initialized. Since patches can be eliminated but not created, the total number of patches in the image decreases with each iteration, while the average patch size increases. These patches have irregular shape, making their boundaries generally hard to notice.

During the first pass of this algorithm, only pixels that were already processed have valid colors and source locations. Therefore, in this pass, a causal context is used both in the comparison and to suggest candidates. Subsequent passes use the whole context to improve the result of the previous passes (using a causal context in these passes would ignore previous passes and create the image afresh each time).

Ashikhmin also showed that this basic algorithm can be slightly modified to perform not only synthesis but also *texture transfer*, where the input image is modified to “look like” the training image. This is done simply by starting with a whole (complete neighborhood) context, and initializing the matrix of source locations to valid random locations in the training image.

Note that super-resolution can be considered as a special case of texture transfer, where the input image is doubled in size by interpolation before “transferring” to it the high frequency details from the training images (at the desired output resolution). If the input image and its corresponding super-resolved version are consecutive layers in a pyramid [19], as they are in an online virtual globe, special care must be taken to guarantee the coherence between the two. To improve the performance of texture transfer with respect to this critical requirement, in a later article, [20], Ashikhmin suggested including, with small probability, an extra candidate with a random source location. These extra candidates provide additional opportunities to match the contexts in the input image, improving the coherence between the input and output images.

Hertzman *et al.*, [21], suggested that to further improve the quality of the results, the source context can be selected by a rule combining both Ashikhmin and Wei-Levoy: choose the best candidate by both methods (q_{ASH} and $q_{W\&L}$), and use q_{ASH} if

$$\|f(C(p)) - f(C(q_{ASH}))\|_2 < k \cdot \|f(C(p)) - f(C(q_{W\&L}))\|_2,$$

with $k > 1$, otherwise use $q_{W\&L}$. The rationale behind this rule is to favor bigger continuous patches unless they diverge “too

⁴ Throughout this article “patches” refer to “connected regions in the output image that are copied verbatim from connected regions in the training image.”

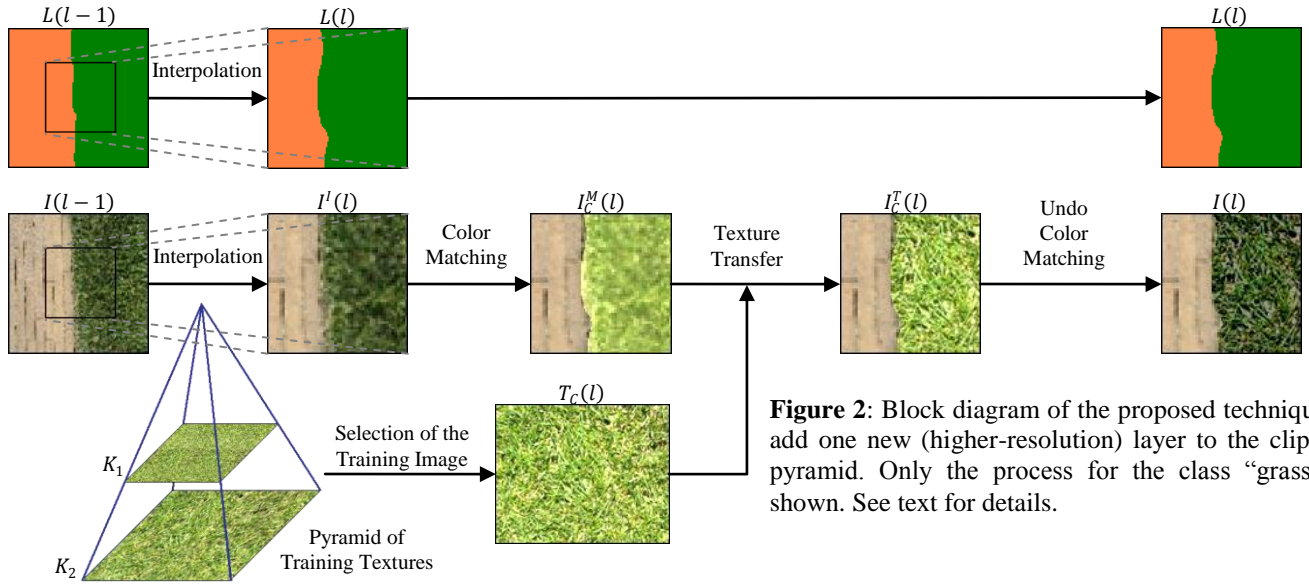


Figure 2: Block diagram of the proposed technique to add one new (higher-resolution) layer to the clipmap pyramid. Only the process for the class “grass” is shown. See text for details.

much” from the input image. This technique produces better results than either Ashikhmin or Wei-Levoy, but has a running time on the order of the slowest of the two (Wei-Levoy). Hertzman *et al.*, building on earlier work by Dalton, [22], showed that rather arbitrary image relationships, including “aesthetic imaging transformations” could be implemented as PSCM.

For completeness, we should mention that two other important image processing operations were also recently shown by Buades *et al.* to be amenable to a PSCM-type implementation: *denoising* [23] and *demosaicing* [24]. These algorithms generalize the PSCM framework by replacing a pixel not with its best match in the training images, but with a weighted average of the best matches. The weight of each pixel being averaged is computed as a decreasing function of a distance between the corresponding contexts.

Contrary to what appears to be widely believed in the community, these algorithms are not sampling from a *Markov Random Field* model of the texture, but from a *Bayesian Network* model, since the processing order of the pixels defines a *directionality* (“causality”) in the link connecting two nodes (pixels) in the graphical model that represents the texture [25].

In the next section we introduce the texture transfer framework we propose for online virtual globe applications. This algorithm runs at least as fast as Ashikhmin, while better maintaining the coherence between the pyramid layers that is important for our interactive application.

3. Proposed super-resolution system

The goal of the proposed algorithm is to “add” new layers at the bottom of the clipmap pyramid used to represent the Earth, beyond the maximum available resolution. This clipmap pyramid contains the lowest resolution image of the Earth in the highest level, an image of double that resolution in the next level, and so on. Layers in the pyramid are aligned so that one pixel is “above” the four (2×2) pixels that represent it at the next (higher) resolution, in the layer below.

The new synthetic layers to be added are not stored in the server and transmitted upon request, but are generated “on the

fly” in the client as needed. The procedure to generate one layer after the last available one, shown schematically in Figure 2, has the following main steps: 1) interpolate the image and class labels of the previous layer to double their resolution; 2) adjust the scale of a training texture to match the scale of the current image; 3) match the colors between the image and training textures to improve the fit of the contexts; 4) transfer the texture from the training textures to the image; and 5) undo the color matching to return the image to its original appearance. This procedure is performed once for each new level “added” to the pyramid. We now provide details for each one of these steps.

3.1. Image and label interpolation

The first stage to generate a new level in the clipmap pyramid is to double the resolution of the image at the previous level, $I(l-1)$, and the labels provided for this (or a higher) image by a contributing user or other “external source,” $L(l-1)$. The contributing user segments the last “real” image in the pyramid into classes by drawing a *rough* curve inside the region corresponding to each class. The system then uses the real-time interactive segmentation algorithm in [16] to obtain the detailed segmentation.⁵

The image’s resolution is doubled by simple bilinear interpolation. To maintain distinct labels separated by smoothly curved borders at the new level, the following procedure is used to interpolate the labels: 1) a mask $M_C(l-1)$ containing the pixels of the class is created for each class C at the previous level; 2) the resolution of each mask $M_C(l-1)$ is doubled, using a Gaussian 2D kernel, obtaining $M_C(l)$; 3) for each pixel in the upsampled image, the class of the highest valued mask is selected as the label of the pixel, $L(l) = \arg \max_C M_C(l)$.

This simple procedure creates region boundaries that seem natural, compared to the artificial “blocky” borders obtained by simple interpolation and thresholding, as illustrated in Figure 3. If desired, these “hard” boundaries can be

⁵ This could be replaced by automatic segmentation techniques, but we restrict ourselves to this semi-automatic approach for the presentation.



Figure 3: Comparison of two algorithms to interpolate the labels, after two consecutive interpolations. Left: interpolation followed by thresholding. Right: our proposed simple approach.

straightforwardly substituted by sigmoidal transitions to avoid aliasing.

This process produces a color image $I^l(l)$ (the superscript “ I ” stands for “Interpolated”) and an image of class labels $L(l)$ of the same size, at the resolution of the new level. The labels $L(l)$ are used to transfer the appropriate texture to each part of the image. The following steps are performed once for each class (label).

3.2. Selection of the training image

Since the contributing user may not have provided a training texture matching exactly the current level, it is necessary to obtain one from the keyframes (training images) that were already provided (K_1, \dots, K_n at keyscales S_1, \dots, S_n). This texture, $T_C(l)$, provides the high frequency details to be transferred, so it can only be obtained by downsampling a higher resolution image K_i , in particular, the first training image having higher resolution than the current level ($S_{i-1} < l \leq S_i$). The maximum zooming level then is given by the lowest keyframe in the pyramid, K_n .⁶

If two consecutive keyframes are separated by more than two levels, the same keyframe (appropriately downsampled) is used to transfer texture to more than one image. This is exploited, as detailed in Section 3.4 below, to further increase the interlayer coherence while improving the intralayer quality.

The downsampling factor is given by the quotient between the current level and the keyscale of the keyframe used (l/S_i). Downsampling is carried out by standard low pass filtering followed by resampling. The output of this process is a training texture $T_C(l)$ at the appropriate scale, for each required class C .

3.3. Color matching

If the (color) histograms of the part of the image that belongs to the class C , $I_C^l(l)$, and the training image $T_C(l)$ used to add the high frequency details, do not match (e.g., due to different illumination conditions during their acquisition), the contexts used in the texture transfer would originate only in rare areas of the training texture (unless the transformation that maps contexts takes this into account, see below). To understand this, consider what would happen if a training texture with light pebbles is used to super-resolve an image with dark pebbles: only the shadow areas between the light

pebbles will be used. This is clearly undesirable.

To address this problem we simply match the mean color of the part of the image belonging to the current class in YIQ colorspace, $\overline{I_C^l(l)}$, with the mean color of the current training texture, $\overline{T_C(l)}$, using the translation $\overline{\Delta} = \overline{T_C(l)} - \overline{I_C^l(l)}$. The output of this process is a color corrected image $I_C^M(l)$ (“ $\stackrel{\text{def}}{=} I_C^l(l) + \overline{\Delta}$ ”) for the part of the image belonging to each class (the superscript “ M ” stands for “color Matched”).

An alternative approach is to include the color compensation in the transformation function $f(\cdot)$ (e.g., by normalizing the norm of the context). We found that this “local compensation” approach tends to match contexts that should not be matched and therefore does not perform as well. In addition, it is slower, since normalization is computed for each context processed. On the other hand, our proposed “global compensation” could fail in cases where there are several textures, with different histograms, mixed in different proportions (e.g., sky with one big cloud and sky with one small cloud), since the global histogram would depend on the mixture proportions. This could be addressed by standard region-based histogram matching techniques [26]. The results in this paper were achieved using only class labels to handle mixtures of textures, and the very simple color-matching scheme described. More sophisticated color matching techniques may provide further refinement.

3.4. Texture transfer

The next step is to transfer, for each class, the texture’s high frequency details from the scaled training texture $T_C(l)$ to the color corrected part of the image $I_C^M(l)$. If this image is the highest in the pyramid that receives texture from a keyframe, this process has three main steps, detailed in Section 3.4.1. Otherwise a different process, presented in Section 3.4.2, is followed.

3.4.1 Texture transfer for the highest level

The first step in the texture transfer procedure for the highest level in the pyramid that receives texture from a keyframe, similar to a Wei-Levoy pass, finds for each pixel in the image a candidate in the training texture that has a similar context. These candidates are used to initialize the matrix of locations Q to valid locations in the training texture, and to copy the colors of those locations from $T_C(l)$ to $I_C^M(l)$.

Wei-Levoy is relatively slow since an expensive tree search is required for every pixel in $I_C^M(l)$.⁷ We avoid this high cost (recall that it is important in our application to synthesize pixels roughly as fast as the server otherwise could provide them), by perceptually-appropriate dimensionality reduction of the contexts. The transformation in this step, $f_1(C_1)$, acts on a 3×3 downsampled context centered at the current pixel (see Figure 4, first row). The contexts extracted from the image in this pass still lack high frequency details, therefore using the original context (not downsampled) would bias the candidate selection towards candidates whose contexts lack high frequencies. We define $f_1(C_1)$ to be the weighted mean

⁷ The performance of a kd-tree degrades rapidly with the dimension of the data, being virtually equivalent to exhaustive search for dimensions greater than 10.

⁶ The lowest keyframe in the pyramid is the highest resolution keyframe.

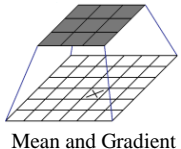
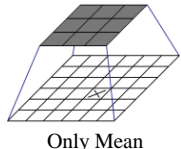
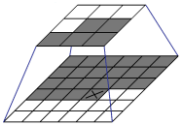
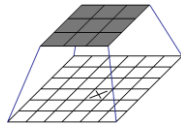
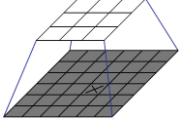
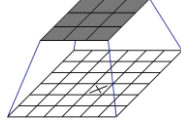
Pass \ Color Channel	Y (luminance)	I and Q (chrominance)
1 st W&L	 Mean and Gradient	 Only Mean
2 nd ASH	 Mean and Gradient	 Only Mean
3 rd and 4 th ASH	 Mean and Gradient	 Only Mean

Figure 4: Context pixels considered in each one of the passes (rows), and each of the color channels (columns). Each context shows two layers: the current level below, and the downsampled current level above (do not confuse with the level above in the clipmap pyramid). Only pixels in gray are considered by the transformation function. The current pixel is marked with an “x”. See text for details.

(in the YIQ colorspace) of the context, concatenated with the gradient of the luminance (Y color channel). Since the dimension of the transformed context is only five (3 + 2 components), tree queries are solved extremely fast, avoiding vector quantization of the texture data, that often compromises the output texture appearance; we use instead a simple kd-tree to answer nearest-neighbor queries. This step produces an image that has the right local structure (colors and edges in the right locations), necessary to enforce interlevel coherence with the upper level image, but still lacks the higher-level structure of the texture.

The next step, an Ashikhmin pass, enlarges the patches, adding the higher-level structure required to increase the intralevel coherence. To do this, the transformation in this step, $f_2(C_2)$, acts on a bigger context, C_2 , that is 6×6 (see Figure 4, second row). The contexts extracted from the image in this pass only have high frequency details in their causal part (the pixels that were already modified). Therefore, using the whole C_2 context would bias the selection towards candidates whose contexts do not have high frequencies in their non-causal neighborhoods. To avoid this, only downsampled, non-causal pixels from C_2 are considered by f_2 . Furthermore, since human subjects are less sensitive to high frequencies in the chroma (I and Q) channels [21] [27], f_2 does not consider high frequency information from the chroma channels of C_2 . Lastly, since humans are not equally sensitive to all color channels [27], we weight the channels of C_2 accordingly. The function f_2 thereby is a concatenation of: 1) the causal part of the Y channel; 2) the non-causal, downsampled, part of the Y channel; and 3) the whole downsampled and weighted I and Q channels.

The next and last step in the texture transfer stage, an additional Ashikhmin pass, adds more details, enlarging the

patches even more. In this iteration, both the causal and non-causal parts of the context have high frequencies. Moreover, since we do not want this pass to overwrite the high frequencies added in the previous pass, the whole context is considered. Hence, f_3 is a concatenation of: 1) the whole Y channel; and 2) the downsampled and weighted I and Q channels (see Figure 4, third row).

Each Ashikhmin pass drives the appearance of the image closer to the training texture, while slowly drifting away from the image in the level above. We found that the best compromise between intra and interlevel coherence was obtained with two iterations of this last step.

3.4.2 Texture transfer for subsequent levels

If for the class C being processed, the current color corrected image, $I_C^M(l)$, and the image above, $I_C^M(l-1)$, receive texture from the same keyframe (appropriately downsampled in each case), the patches in the image above can be *directly* super-resolved, avoiding the first step of the texture transfer algorithm altogether. Recall that a patch R_{l-1} in $I_C^M(l-1)$ is (by definition) copied verbatim from $T_C(l-1)$. Since $T_C(l-1)$ and $T_C(l)$ were both downsampled from the same keyframe, there exist a super-resolved version of R_{l-1} (R_l) in $T_C(l)$. Therefore, R_l can be used to super-resolve R_{l-1} , by simply propagating the locations inside R_{l-1} in the layer above to R_l in the layer below (transformed by a simple formula to account for the change of scale).

This process guarantees that the coherence between levels is completely preserved within patches (since R_{l-1} was downsampled from R_l), but does not guarantee a seamless integration between patches in the same level. To mask the seams between these patches, candidates suggested in the first step above are used to fill-in pixels that lie next to a patch boundary (see Figure 5). This increases the number of candidates considered near patch boundaries, making more likely that a candidate that better masks the seam would be found. In addition, this process creates increasingly larger patches (note that R_l has four times more pixels than R_{l-1}), as subsequent images receive texture from the same keyframe (more on this in Section 4 below), further improving the global appearance of the current level.

To conclude this section, the proposed texture transfer algorithm combines small patches in the highest level receiving texture from a keyframe, to ensure that the global

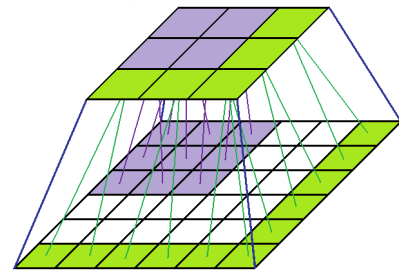


Figure 5: Propagation of the locations in two patches (in green and violet) to the next level. The locations in the colored pixels of the lower level are computed from the corresponding locations in the upper level. The locations in the white pixels, lying near the patch boundary, are initialized using a tree search as detailed in the text.

appearance of the previous level is preserved. In each subsequent level, the patches are enlarged, and their boundaries are refined to better hide the seams.

3.5. Undo color matching

Before storing the final image in the new (deepest) level of the clipmap pyramid, the color transformation has to be undone to restore the original appearance of the image, matching the histograms of the levels above.

This concludes the description of the proposed method. In the next section we show results obtained with it.

4. Results

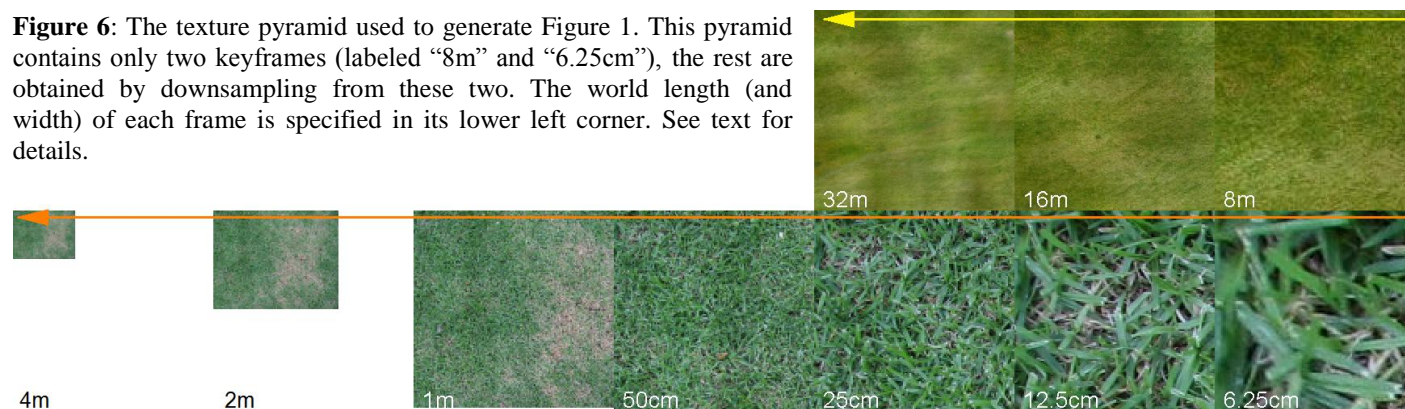
To illustrate the proposed framework, we now present some examples. The reader is encouraged to check the movies at [28] in order to see additional examples and fully appreciate the technique.

In Figure 1, the proposed framework was used to add details to the grass of the Maracanã stadium in Rio de Janeiro, Brazil. The first four images, extracted from Google Earth, are shown unchanged and represent the maximum resolution available in Google Earth for this part of the world. The rest of the images were synthesized by super-resolving the center square of the previous image (dashed), following the framework here proposed. The real world size of the images (stated in the lower-left corner) is halved in each step. Note that the general appearance of the previous image is respected, while the texture is transferred to each image; in particular look at the grass tone bands in the fifth image.

Figure 6 shows the texture pyramid, with only two keyframes, used to synthesize the images in Figure 1 (no texture was transferred in the first four images). As detailed in Section 3.2, these two keyframes are downsampled to produce the texture at all the levels required by this example. When a keyframe is downsampled by 2 (to obtain a texture one whole level higher in the pyramid), the number of pixels in the training texture is reduced by four. Therefore, having two keyframes separated by n levels implies that it may be necessary to downsample the lower texture by 2^{n-1} . The keyframe then must be large enough to allow a 4^{n-1} -fold reduction in size, while still being large enough to serve as a training texture (this explains why the images labeled “4m” and “2m” are smaller in Figure 6). We found it impractical to have keyframes more than 6-7 levels apart.

The original location, in the training texture, of each pixel

Figure 6: The texture pyramid used to generate Figure 1. This pyramid contains only two keyframes (labeled “8m” and “6.25cm”), the rest are obtained by downsampling from these two. The world length (and width) of each frame is specified in its lower left corner. See text for details.



transferred to the images in Figure 1 is shown in Figure 7. The vertical and horizontal pixel indices are encoded in the red and blue channels of the images, respectively. Note how the size of the patches increases in the sequence of images with texture transferred from the same keyframe. In particular, note that the last image of the sequence was almost transferred “in one piece.” The rest of the images were assembled using many patches, yet the seams between patches are virtually impossible to notice. This phenomenon of increasing patch size suggests the use of keyframes as separated as possible, subject to the upper limit mentioned above.

All the images in this example contain 256×256 pixels, and each one of them was generated in 1 to 3 seconds (depending on the average patch size at the level and the size of the training texture), in a 1.8Ghz Turion machine. The algorithms, implemented part in C++ and part in Matlab, are not optimized for speed but for ease of experimentation. If desired, a speedup of at least an order of magnitude can be achieved, or even more if exploiting a GPU.

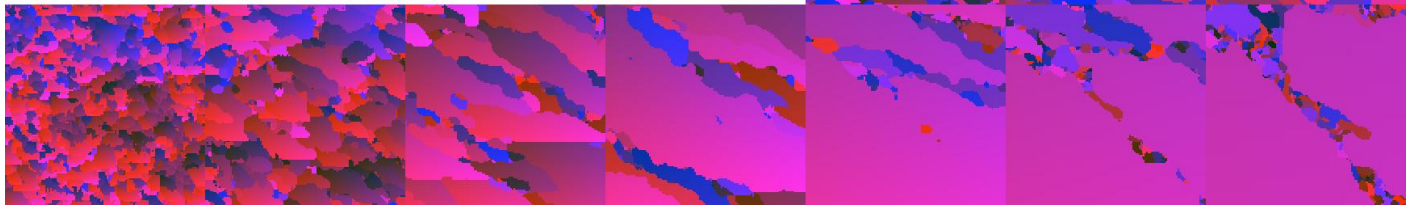
An additional example in a rural setting is included in Figure 8. Given the abundance of easily-segmentable large uniform regions, this setting is ideal for our framework.

5. Conclusions and future work

In this work we introduced an approach to satisfy the user’s constant demand for higher resolution images in virtual globe applications. The proposed framework reduces the operational cost of such Internet applications, while at the same time improving the quality of the displayed images and augmenting their information content. This is achieved by state-of-the-art image processing techniques.

The proposed framework can be further improved or extended in a number of ways. First, the coherence between layers of the pyramid can be increased, while reducing the on-line computing cost, by off-line pre-computing a list of appropriately downsampled similar neighborhoods between consecutive keyframes of the texture pyramid. Since this step is carried out off-line, these neighborhoods can be larger than the ones used in this work, and an increase in the coherence between layers is expected. This step is similar to the analysis phase proposed in the Jump Map method [29], although in this case, similar neighborhoods are computed between textures in consecutive pyramid keyframes rather than within the same texture.

Figure 7: The location, in the training texture, of each pixel copied to the images in Figure 1. The vertical and horizontal pixel indices are encoded in the red and blue channels respectively. The size of the patches increases in each sequence of images with texture transferred from the same keyframe.



Secondly, the boundaries between classes currently produced by the proposed approach can be abrupt (see the videos at [28]), misrepresenting the distinctive transitions actually observed between two particular classes (e.g., grass and sand, or sand and sea). A better way to handle these transitions is to include exemplars of them in the texture training set, and use the texture transfer algorithm for their reconstruction. As mentioned in Sec 3.3, mixtures of textures are not satisfactorily handled by the simple color matching essayed here, which must be replaced by a more sophisticated algorithm to pursue this approach.

Thirdly, the proposed framework could be extended to generate the texture pyramids needed to render textured surfaces at different scales in video games and virtual worlds. Essentially, the proposed framework would act like a procedural texture generator [30], where textures are defined and controlled directly and intuitively by the keyframe images rather than indirectly, by other than visual means. Results in these directions will be reported elsewhere.

Acknowledgements

This work was mostly performed while DR was an intern at Google, Inc. We thank Google, Inc. for this support. Additional support came from NSF, ONR, NGA, ARO, and DARPA.

6. References

[1] Wikipedia., Virtual globe. Online:http://en.wikipedia.org/wiki/Virtual_globe, 2008.
 [2] Google, Inc., Google Earth. Online: <http://earth.google.com>, 2007.
 [3] Google, Inc., Google Maps. Online: <http://maps.google.com>, 2007.
 [4] Microsoft Corp., Microsoft Virtual Earth™. Online: <http://dev.live.com/virtualearth/>, 2007.
 [5] Yahoo! Inc., Yahoo! Maps. Online: <http://maps.yahoo.com>, 2008.
 [6] Motherplanet, Inc., Motherplanet. Online: <http://www.motherplanet.net>, 2007.
 [7] NASA., World Wind. Online: <http://worldwind.arc.nasa.gov>, 2007.

[8] Tanner, C., Migdal, C. and Jones, M., "The clipmap: A virtual mipmap." *SIGGRAPH*, 1998.
 [9] C.I.A., "The World Factbook." Online: <https://www.cia.gov/library/publications/the-world-factbook/>, 2008.
 [10] Food and Agriculture Organization of the United Nations., *FAO Production Yearbook 1994*, Rome, Italy, 1995.
 [11] National Agricultural Statistical Service., Online: <http://www.nass.usda.gov>, 2007.
 [12] Wei, L. and Levoy, M., "Fast Texture Synthesis using Tree-structured Vector Quantization." *SIGGRAPH*, 2000.
 [13] Ashikhmin, M., "Synthesizing Natural Textures." *ACM Symposium on Interactive 3D Graphics*. 2001.
 [14] Koriakine, A. and Saveliev, E., WikiMapia. Online:wikimapia.org, 2008.
 [15] Google, Inc., Sketchup. Online: <http://www.sketchup.com>, 2007.
 [16] Bai, X. and Sapiro, G., "A geodesic framework for fast interactive image and video segmentation and matting." *ICCV*, 2007.
 [17] Freeman, W. T., Jones, T. R. and Pasztor, E. C., "Example-Based Super-Resolution." *IEEE Computer Graphics and Applications*, 2002.
 [18] Efros, A. A. and Leung, T. K., "Texture Synthesis by Non-parametric Sampling." *ICCV*, 1999.
 [19] Williams, L., "Pyramidal parametrics." *SIGGRAPH*, 1983.
 [20] Ashikhmin, M., "Fast Texture Transfer." *IEEE Computer Graphics and Applications*, 2003.
 [21] Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B. and Salesin, D. H., "Image Analogies." *SIGGRAPH*, 2001.
 [22] Dalton, J., "Adaptive Learning of Aesthetic Imaging Transformations." *Conference Proceedings Digital Image Computing: Techniques and Applications, Sidney, Australia*. 1993.
 [23] Buades, A., Coll, B. and Morel, J. M., "Image and movie denoising by nonlocal means." *IJCV*, 2006.
 [24] Buades, A., Coll, B., Morel, J. M. and Sbert, C., "Non local demosaicing." *IEEE TIP*, 2007.
 [25] Bishop, C.M., *Pattern Recognition and Machine Learning*. Springer, 2006.
 [26] Caselles, V., Lisani, J. L., Morel, J. M. and Sapiro, G., "Shape Preserving Local Histogram Modification." *IEEE Trans. Image Proc*, 1999.
 [27] Wandell, B. A., *Foundations of Vision*. Sinauer Associates, Inc., 1995.
 [28] Rother, D., Super-Resolution Texturing. Online: www.diegorother.net/Research/SuperResolutionTexturing.html, 2008.
 [29] Zelinka, S. and Garland, M., "Towards real-time texture synthesis with the Jump Map." *Eurographics Workshop on Rendering*, 2002.
 [30] Wikipedia., Procedural texture. Online: http://en.wikipedia.org/wiki/Procedural_texture, 2007.

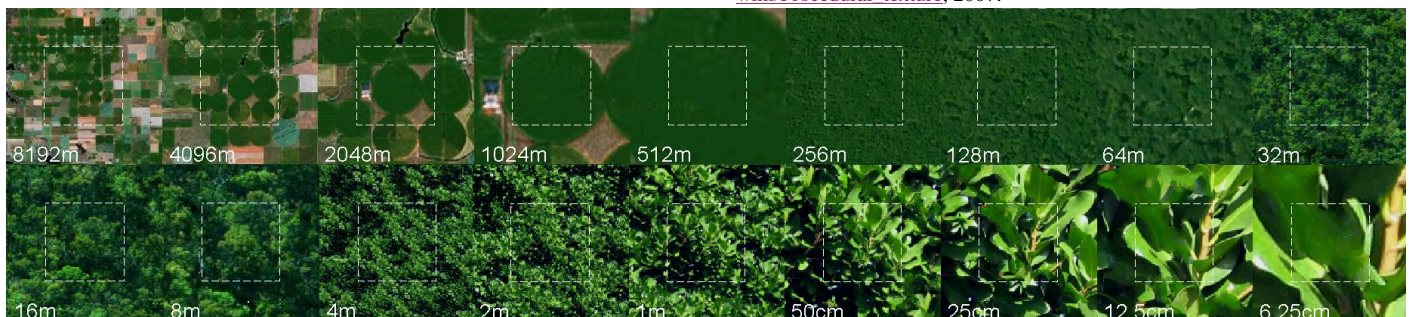


Figure 8: Example in a rural setting, a field in Iowa. See the corresponding video at [28].