Fast Image-based Rendering using Hierarchical Image-based Priors

Oliver Woodford
Department of Engineering Science
University of Oxford
ojw@robots.ox.ac.uk

Andrew W. Fitzgibbon
Microsoft Research
Cambridge, UK
awf@microsoft.com

Abstract

Novel view synthesis using image-based priors has recently been shown to provide high quality renderings of complex 3D scenes. However, current methods are extremely slow, requiring of the order of hours to render a single frame. In this paper we show how a coarse-to-fine method can be used to reduce this time significantly. In contrast to traditional multiple-view stereo methods, devising a coarse-to-fine strategy for this problem is complicated by the fact that image-based priors are strongly tied to the scale at which rendering is performed. We show how a hierarchical decomposition of the texture patch database both allows multiple-scale analysis and speeds up the imposition of the priors. Examples are shown on a number of challenging sequences, and illustrate that the new method yields comparable results to the previous method, with significant gains in speed.

1 Introduction

The new-view synthesis (NVS) problem may be stated as follows: given a set of images of a 3D scene with corresponding camera positions, generate a view of the scene from a new viewpoint, not in the original set. Approaches to the NVS problem can be roughly subdivided into those that explicitly create a 3D representation of the scene from which all new views are rendered, and those that do not. Methods in the first class generally create volumetric (voxel) representations [3, 15] or texture mapped mesh representations of the scene [11]. The second group of methods generally comes under the umbrella of image-based rendering [6, 7, 16]. The aim of such methods is to generate photorealistic renderings of a scene based directly on a set of input images, without an explicit reconstruction of 3D depth. A problem which is shared by all NVS systems is the inherent ambiguity of the problem—there are many 3D scenes which could have given rise to a given set of input images, and thus there are many equivalent synthesized images which are consistent with a given input set. To choose between the many nearly equivalent solutions requires strong and accurate priors on the class of scenes to be rerendered.

Recently, Fitzgibbon *et al.*[5] have introduced an *image-based* prior to the NVS problem. While previous work had essentially modelled the scene as piecewise smooth, the image-based prior represents the constraint that the synthesized image should have the same local statistics as the input images. This allows realistic rendering of scenes containing fine detail such as hair and textured surfaces, which previous methods tended to blur. Their approach minimizes an energy which contains two terms: one measures the

a b c

Figure 1: **Steadicam sequence.** (a) An input image from the monkey sequence [5]. (b) A novel view from [5] (extracted from compressed video), with a quoted rendering time of 4.4 hours. (c) Our rendering from a viewpoint as close as possible to the original. The results are better under the monkey's right arm and comparable elsewhere, with a rendering time of 93 seconds.

photoconsistency between the synthesized and input views, while the second measures the similarity of each small (e.g. 5×5 pixels) patch in the synthesized view to the most similar patch in the input sequence. This effectively encourages the output images to be rendered as a patchwork of small windows from the input sequence. This produces rendered sequences of very high visual quality, but has the disadvantage that it is extremely slow (the paper quotes rendering times of 10 pixels per second, or of the order of 11 hours per PAL frame.)

In this paper we show how to significantly improve the speed of the algorithm using a multiresolution strategy. This is a nontrivial extension of existing multiresolution approaches to stereo, because the texture prior is not scale-invariant. Our contribution is both to extend current methods for high-dimensional nearest-neighbour search, and to use the particular characteristics of image patches to increase the efficiency of patch search. As Figure 1 shows, our new approach yields results of at least the same quality as the earlier work, with a considerable increase in speed.

1.1 Related work

Two strands of research relate to this problem: multiview dense stereo and nonparametric texture synthesis.

Multiresolution approaches to dense stereo computation have been common since at least the work of Marr and Poggio [14]. The basic strategy is to solve for depths or disparities in a coarse-scale low-resolution image, and use those disparities to seed search at the next finer scale. Adapting this approach to multiview stereo, as introduced by Okutomi and Kanade [18], is straightforward, but is discussed below for completeness.

The second strand of relevant research is in texture synthesis: the generation of synthetic images which have local statistics similar to a supplied example image. Recent work largely builds on the patch-based approaches of Efros, Leung and Wei [4, 20], which give results of startling visual quality using very simple algorithms. The core of these al-

gorithms is a large database of texture patches—generally every $W \times W$ subwindow of the example texture image, where typical values of W are from 5 to 15 pixels. Texture synthesis amounts to a nearest-neighbour search in this database for every generated pixel.

Early implementations were extremely slow, performing a linear search for the best matching patch. Wei and Levoy [21] introduced tree-structured vector quantization (TSVQ), with considerable improvements in speed, but at the cost of reduced quality of synthesis. Liang *et al.*[12] assess TSVQ and observe that this reduced quality stems from the fact that TSVQ's greedy search, though fast, can return matches which are quite dissimilar to the query patch. (Liu *et al.*[13] dub this strategy "defeatist search"). Liang *et al.* replace TSVQ with a kd-tree [17], which overcomes the problems induced by defeatist search. When adapting their method to multiscale, they simply use a quadtree decomposition. However, to avoid boundary artifacts they essentially compute multiple quadtrees, sliding the inter-node boundaries across the image, increasing the storage requirements by a factor of more than 16. Returning to defeatist search, Liu *et al.*[13] observe that it can be an order of magnitude faster than search in a kd-tree, and that its failures can be overcome by using not a tree but a singly rooted directed graph, which they call a "spill tree". Spill trees are kd-trees which allow overlapping nodes, so that points near a node's decision boundary are assigned to both children of the node. In this paper, we show how a multiresolution analogue of the spill tree allows fast and accurate patch search. The advantages over Liang *et al.* are reduced memory usage and simpler tree-construction and search algorithms. We shall use the incorrect term "tree" to describe this data structure, thinking of it as a tree containing duplicated subtrees.

In the following sections we shall formally define the NVS problem, expressing it as an energy minimization. We then describe our multiresolution scheme without texture priors. In Section 5 we describe our new texture patch search algorithm, and show how it may be integrated into the multiresolution depth estimation. We conclude by demonstrating the speedups obtained by the new technique.

2 Problem statement

We are given as input a set of 2D images of a 3D scene, I_1 to I_n, each of which has an associated 3×4 projection matrix $\mathtt{P}_i$ defining the i^{th} image's camera position [9]. We are given a new camera position, not in the original set, defined by a projection matrix $\mathtt{P}_o$. The task is to produce the output image $\mathcal{V}$, which is the view of the scene from the new camera position. The image $\mathcal{V}$ comprises pixels $V(x,y)$, the colour, expressed as a vector in the appropriate colour space, at pixel (x,y). We are at liberty to assume that world coordinates have been transformed so that the camera centre is the origin ($\mathtt{P}_o = \mathbf{0}$) and that the 3D ray corresponding to new-view pixel (x,y) is $\mathbf{X}(z) = [xz, yz, z, 1]^\top$. Let π be the perspective division function $\pi(x,y,z) = (x/z, y/z)$. Array accesses of the form $I(\mathbf{u}), \mathbf{u} \in \mathbb{R}^2$ are assumed to be bilinearly interpolated.

Following [5], NVS is expressed in a Bayesian framework of *photoconsistency* [19] and image-based priors, so that producing an output image is posed as the minimization of the negative log posterior, or "energy", given by

$$E(\mathcal{V}) = E_{\text{photo}}(\mathcal{V}) + \lambda E_{\text{texture}}(\mathcal{V}). \tag{1}$$

The first term, E_{photo}, measures the photoconsistency at each pixel between the input images and the new-view pixel. The second measures the extent to which the new view

is "image-like", with a tuning variable, λ, that weights the importance of E_{texture} relative to E_{photo}. We take the photoconsistency energy to be the sum of the mean of the squared distances of the input samples from the rendered value. Thus

$$E_{\text{photo}}(\mathcal{V}) = \sum_{x,y} \min_{z_{min}<z<z_{max}} \frac{1}{n} \sum_{i=1}^{n} \|V(x,y) - I_i(\pi(\mathtt{P}_i \mathbf{X}(z)))\|^2. \tag{2}$$

Note that we explicitly minimize over depth at each pixel to find the most likely colour, and that minimizing E_{photo} alone is equivalent to the NVS strategy of Irani *et al.*[10]. The texture energy is the sum over all pixels of the distance of the pixel's neighbourhood to the closest patch in the input images, defined as

$$E_{\text{texture}}(\mathcal{V}) = \sum_{x,y} \min_{\mathbf{T}\in\mathbb{T}} \|\mathtt{W}(\mathbf{T} - \mathbf{N}(\mathcal{V};x,y))\|^2 \tag{3}$$

where $\mathbf{N}(\mathcal{V};x,y)$ is some vector of values from the neighbourhood of $V(x,y)$. If we are considering 5×5 windows of RGB images then $\mathbf{N} \in \mathbb{R}^{75}$. The *patch library* $\mathbb{T}$ is the collection of all patches in the input sequence:

$$\mathbb{T} = \{\mathbf{N}(I_k;x,y) \mid \forall\, x,y,k\}$$

Finally, $\mathtt{W}$ denotes a normalised diagonal weighting matrix that can be used to vary the importance of different values in the neighbourhood, such as the Gaussian kernel used in [4].

3 Computational strategy: single resolution

To find the $\mathcal{V}$ that globally minimizes $E(\mathcal{V})$ would require a search over the entire space of rendered images, $\mathbb{R}^{colours \times width \times height}$. This is a huge search space. In order to reduce the search space our approach is to optimize over depth, finding z at each pixel within an iterated conditional modes (ICM) framework [2]. We describe our strategy at a single resolution before proceeding to the new multiresolution algorithm. This strategy differs from previous work in that it explicitly searches over depth (instead of colour), and more accurately includes the texture energy.

3.1 Minimizing $E(\mathcal{V})$ over depth

Minimization over depth may be performed at each pixel independently, but, as the texture stage will not be completely independent of the depth stage, we benefit from caching some information. We compute a 3D array, $\mathcal{C}$, that contains a colour for every output pixel at every depth, that colour being the mean of the input-image pixels to which the 3D point at (x,y,z) projects. Specifically,

$$C(x,y,z) = \frac{1}{n} \sum_{i=1}^{n} I_i(\pi(\mathtt{P}_i \mathbf{X}(z))). \tag{4}$$

Then E_{photo} may be precomputed at each point in $\mathcal{C}$. We then find the array of depths, $\mathcal{Z}$, that minimizes (1), where $V(x,y) = C(x,y,Z(x,y))$. We do this by using ICM to find a local minimum.

Let the depth map at iteration t be $\mathcal{Z}_t$. We obtain the initial estimate, $\mathcal{Z}_0$, as the minimizer of $E_{\text{photo}}(\mathcal{C}(\mathcal{Z}_0))$, and continue iterations until $E(\mathcal{C}(\mathcal{Z}_t)) >= E(\mathcal{C}(\mathcal{Z}_{t-1}))$. If the current depth map estimate is $\mathcal{Z}_t$, then the next is obtained by parallel update at each pixel (x,y) to compute $\mathcal{Z}_{t+1}$:

$$Z_{t+1}(x,y),\, L(x,y) = \underset{z_{\min}<z<z_{\max},\; l|\mathbf{T}_l\in\mathbb{T}}{\operatorname{argmin}} E_{\text{photo}}(C(x,y,z)) + \lambda\|\mathtt{W}(\mathbf{T}_l - \mathbf{N}(\mathcal{C},\mathcal{Z}_t;x,y,z))\|^2 \tag{5}$$

In this minimization we also record, for each pixel, the index, $L(x,y)$, of the texture patch which minimizes the energy of that pixel. We use 5×5 windows to create the pixel neighbourhood vectors, and all pixels in the neighbourhood have the same weighting:

$$\mathbf{N}(\mathcal{C},\mathcal{Z}_t;x,y,z) = \{\{C(x+i,y+j,Z_t(x+i,y+j)) \mid -2\le i,j\le 2,\; i,j\neq 0\},\; C(x,y,z)\}$$

Since we already have precomputed values for E_{photo} for every colour in $\mathcal{C}$, we can optimize our search to finding the texture patch, $\mathbf{T}$, that minimizes $E(\mathcal{V})$, taking the E_{photo} values at Z_t for each pixel other than the current update pixel (x,y). The computational cost of this stage is dominated by the need to perform a closest-patch search in $\mathbb{T}$ for every pixel, which will be accelerated in §5.

On convergence, each pixel of our output image, $\mathcal{V}$, is taken to be the centre pixel of energy minimizing texture patch found in (5):

$$V(x,y) = \mathbf{T}_{L(x,y)00}$$

For notational convenience, $\mathbf{T}_{00}$ denotes the element of vector $\mathbf{T}$ corresponding to the centre pixel of the neighbourhood it represents. For a 5×5 patch, this would be element 13 of $\mathbf{T}$. The image given by $\mathcal{C}(\mathcal{Z}_\infty)$, where $\mathcal{Z}_\infty$ denotes the energy minimizing $\mathcal{Z}$ found in (5), is made up of averaged, bilinearly interpolated samples of input images, and will therefore have lost some high frequency detail. The final stage of our rendering process constrains the colours in our output image to be in the set of colours seen in the input images, with no relation to the elements of $\mathcal{C}$. This enables the output image to retain both the high frequency details and the purity of colours seen in the input images.

Implementation issues For each output image we use only the 8 closest input images, in terms of the euclidean distance between input and output camera centres. We choose a selection of sampling depths such that the epipolar lines in our input images are optimally covered, with a minimum distance between sample points of half a pixel. This reduces our initial sample set to the bare minimum, though without a multiresolution strategy it is still necessary to sample of the order of 50–100 depth values at each pixel.

4 Multiresolution implementation

By posing our algorithm as the evaluation of depth rather than colour we enable the use of scale-space to reduce rendering time. The slowest part of our algorithm is the minimization of energy over depth. Since ICM convergence time is a function of the number of labels (i.e. depth values), reducing the number of labels speeds up rendering. We can reduce the number of depth labels by iteratively refining the depth estimate of each pixel

in a coarse-to-fine manner. In order for our photoconsistency energy at the coarser scale to represent the photoconsistency over a wide range of depths we must low-pass filter our input images before sampling. The size of low-pass filter depends on the number of fine depth levels, s, between depth samples at our given scale. Working on the basis that there is a maximum of half a pixel between fine level depth samples when projected into the input images, we use a disk shaped averaging filter with radius $s/2$ to low-pass filter the input images. Our image-based texture priors must be similarly low-pass filtered. We minimize $E(\mathcal{V})$ over depth at each scale, then propagate the depth through to the next scale, where we refine our search around that depth. If we normally sample at 50 depth levels then 3 passes of 7 depth levels each adequately covers the range.

The time taken to calculate each ICM iteration is linearly proportional to the number of pixels, so a further and much greater speed up comes from calculating the coarser scales at a lower resolution. We downsample every coarse scale by a factor of 2 from the previous scale, which, for 3 scales, makes the coarsest scale approximately 16 times faster to calculate. Again, the texture priors need to be similarly downsampled. A downside of this method is that at each jump in scale we must estimate depths for 3 out of 4 points, since we are upsampling by interleaving. However, simply bilinearly interpolating depths from the points we know appears to produce reasonable results. All timings quoted in the paper use this downsampling, but, for convenience and simplicity of exposition, we assume that *the images at all scales have the same size*. The only difference between input images at different scales is the amount of low-pass filtering they have undergone. We shall use superscript S to indicate the scale at which an entity is defined, with the finest scale corresponding to $S = 1$.

5 Speeding up texture patch lookup

Recall that the patch library, $\mathbb{T}$, is a collection of all patches in all our input images (including overlaps). For 11 input images of size 800×600 pixels that is approximately $m = 5 \times 10^6$ patches. Because we are working at multiple scales, we have one library, $\mathbb{T}^S$, for each scale S.

If we denote the texture patch library by $\{\mathbf{T}_l\}_{l=1}^{M}$ then patch lookup is a weighted search in a d-dimensional point set, $\mathbb{T}$, where $d = 75$ for 5×5 RGB patches. Thus, at a given scale we wish to efficiently compute the closest point to a query point, $\mathbf{N}$, written

$$\min_l \|\mathtt{W}(\mathbf{N} - \mathbf{T}_l)\|^2$$

This is speeded up in two ways. First, the patches are clustered into a smaller set of key patches, which provides the greatest acceleration at the coarsest scale; then the relationship between the patches at different scales is used to define a hierarchical data structure which accelerates patch search at all scales but the coarsest.

Patch clustering Particularly at coarser scales, many of the patches in the library are very similar to each other. Thus, we can reduce the library size by clustering so that all patches in the library are within a threshold similarity of at least one cluster centre. For a given patch library, $\mathbb{T}$, denote the new clustered set by $\mathbb{U}$, containing cluster centres $\mathbf{U}_{1..m}$. Define $u(\mathbf{T}) = \text{argmin}_k \|\mathbf{T} - \mathbf{U}_k\|$, i.e. the index of $\mathbf{T}$'s closest cluster centre. Ideally, given

	All depths	Coarse-to-fine
Without image priors	140	10
With image priors	4695	1079
	(a) Plant	

All depths	Coarse-to-fine
93	7
1312	93
(b) Monkey	

Table 1: **Timings**. Times, in seconds, taken to render each image. In both cases, stereo without texture priors is faster by a factor of about 13. When patch priors are included, the monkey retains this factor, while the plant image has less of a speedup (a factor of 4.3) because of the large textured green area in the background, which reduces the effectiveness of the hierarchical method.

a threshold distance τ, we would compute the *smallest* $\mathbb{U} \subset \mathbb{T}$ for which

$$\min_k \|\mathbf{T} - \mathbf{U}_k\| < \tau \qquad \forall\, \mathbf{T} \in \mathbb{T}.$$

In practice this is an intractable problem, but a reasonably small $\mathbb{U}$ (say no more than twice as large as the minimum possible) can be found using relatively simple algorithms. We use Hartigan's sequential leader clustering (SLC) [8]. At the finest scale, we set the threshold to correspond to an RMS of 0.7 grey levels (i.e. $\tau = 10.5$ for 5×5 RGB patches). At the coarse scales, where high-frequency detail is less important, an RMS of 1.2 grey levels gives a smaller $\mathbb{U}$.

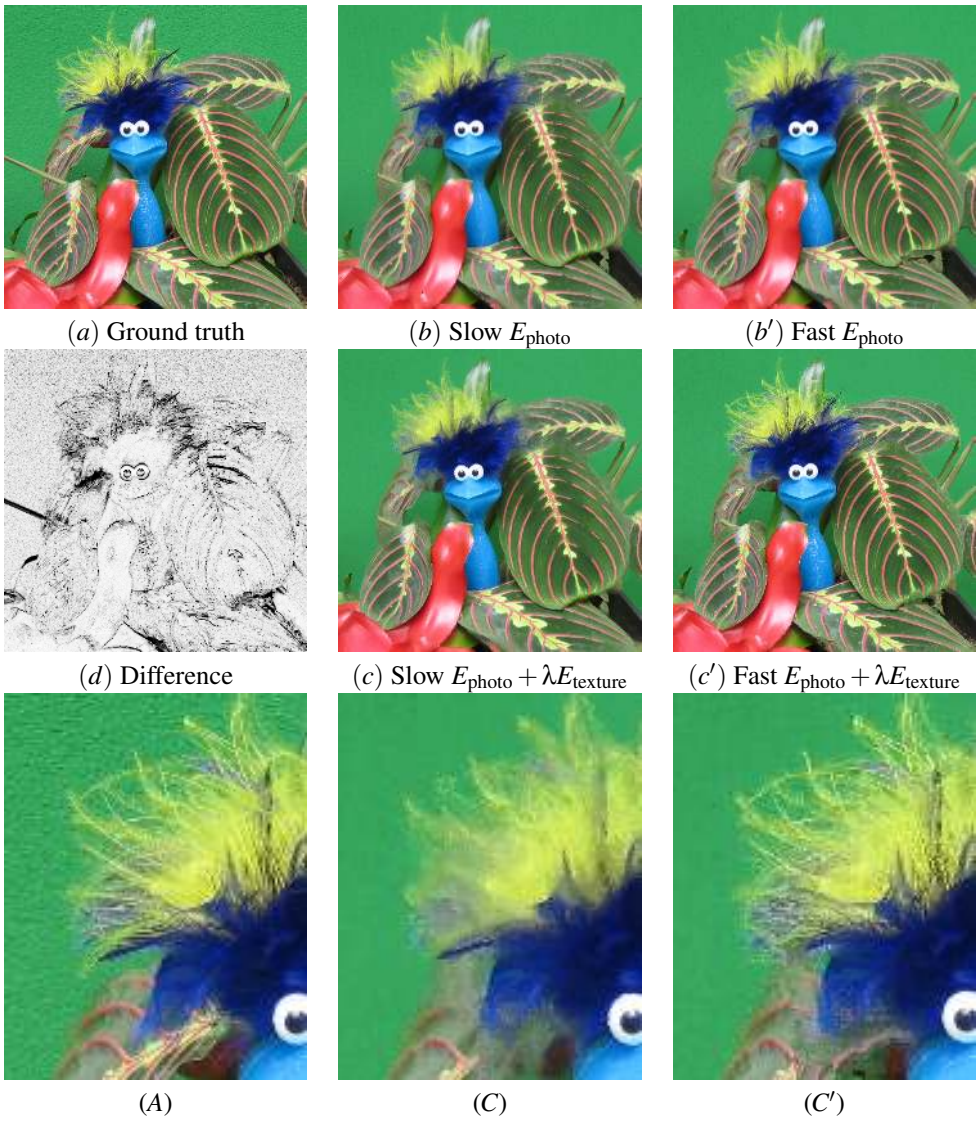
Multiscale patch hierarchy We now come to the source of the most substantial speedup in the implementation. Although patch clustering reduces the size of the patch libraries, the requirement for relatively small thresholds τ means that the speed improvement is rarely greater than a factor of four or five. However, after the coarsest scale we can obtain a greater speedup from a hierarchical representation, which we shall now describe.

At every scale but the finest, we associate with each patch (i.e. cluster centre) a list of "child" cluster centres in the next finer scale. If we denote a patch by $\mathbf{U}'_{xy}$, and the patch at the next finest scale centred on[1] the same pixel location (x, y) in the texture source image by $\mathbf{U}_{xy}$, then each $\mathbf{U}'_{xy}$ is linked to the set of $\mathbb{U}$ whose similarity is within a threshold β of $\mathbf{U}_{xy}$. Then, when moving from coarse-to-fine in the multiresolution implementation, optimization of (5) at each pixel searches only the children of the patch which minimized (5) at the coarser scale. In our implementation, with β corresponding to an RMS of 9.6 grey levels, the average size of the child lists[2] on the "plant" image was 149 at the finest scale and 4.4 at the next finer scale.

At the coarsest scale we have no patch index from the previous scale, therefore we must search through all the quantized patches for every pixel at this scale. Fortunately, there are fewer pixels as a result of downsampling, and fewer patches because of the clustering.

[1] Note that, when coarser scales are downsampled, $\mathbf{U}_{xy}$ and $\mathbf{U}'_{xy}$ will cover different sized areas of the texture source image.

[2] When coarser scales are downsampled, some pixels don't have associated child lists. Their child lists are created by concatenating the known child lists of neighbouring pixels. The patch centres from each neighbour's child list are offset by the distance from the neighbour to the pixel in question prior to concatenation.



(a) Ground truth (b) Slow E_{photo} (b') Fast E_{photo}

(d) Difference (c) Slow $E_{\text{photo}} + \lambda E_{\text{texture}}$ (c') Fast $E_{\text{photo}} + \lambda E_{\text{texture}}$

(A) (C) (C')

Figure 2: **Leave-one-out test.** (a) Ground-truth image, one of twelve from the "plant" sequence. $(b), (b')$ Rendered images without texture priors. $(c), (c')$ Rendered images with texture priors. Note the improvement in detail in the hair and on the blue stem of the toy. (d) Difference between (c') and the ground truth showing how, in common with [5], the coarse geometry is consistent with the true new view, while fine details are essentially invented by texture synthesis. Bottom row: zooms of a, c, c'.

6 Examples

One image sequence was obtained from Fitzgibbon *et al.*for direct comparison with their results, and a further sequence was captured using a digital stills camera and calibrated using commercially available software [1] to give us the projection matrices.

Our first experiment is a leave-one-out test. We reconstruct each of the 12 input images of our "plant" sequence using the other 11 images. Taking a typical image, Figure 2 shows the ground truth, as well as 4 output images created by minimizing $E_{\text{photo}}(\mathcal{V})$ and $E(\mathcal{V})$ with and without our coarse-to-fine algorithm using hierarchical texture priors. Comparing the two output images (b) and (b') that don't use texture priors (i.e. just minimize E_{photo}), we can see the effect of using scale-space clearly. The artifacts in (b) are less widespread than in (b'), as the spatial support at coarse scales aids depth estimation, but each artifact covers a larger area, as the wrong choice of depth for a single pixel at a coarse scale affects multiple pixels at a fine scale. Comparing (c) with (b) shows some quality improvements on edges, while other details become more blurred. By contrast, comparing images (c') with (b') we can clearly see the effect of using image-based texture priors. The image using texture priors is sharper, bringing out the high frequency detail in the leaves and feathers, and reproducing specularities on the body of the toy. This improvement in image quality over the non-hierarchical image (c) results from the coarser scales constraining the patch search at the finer scales, and shows that the new algorithm can give higher quality as well as faster results. However, the hierarchical method still fails to reproduce the high frequency detail of the baize background.

Our second experiment is a recreation of the steadicam monkey video produced in [5]. As Figure 1 shows, our rendering of the monkey is of comparable quality, with improvements in the background reconstruction resulting from our use of only a few close input images rather than all the images. However, the real improvement comes from the two orders of magnitude increase in rendering speed.

7 Conclusion

This paper has presented a scheme for the acceleration of new-view synthesis using patch priors. We introduced an image-priors-based NVS technique that minimizes energy over depth, rather than colour as per [5]. On its own this method speeds up rendering time by an order of magnitude, but at the cost of image quality. However, we have shown how posing the problem in this way allows us to leverage scale-space not only to reduce the number of depths sampled, but also to constrain the patch search at a given scale according to the result of energy minimization at the previous scale. This both reduces rendering time by a further order of magnitude and increases image quality, showing that by introducing a hierarchical representation integrated with a multiscale search, considerable speed improvements are possible over existing schemes, with equivalent or higher quality. We have shown the main failure mode of the algorithm—large areas of the image where texture exists only at the finest scale do not benefit from the clustering at coarse scales. We hope to repair this deficiency by more intelligent scale selection, and by the use of a hybrid of the hierarchy and a kd-tree.

References

[1] 2d3 Ltd. http://www.2d3.com, 2002.

[2] J. Besag. On the statistical analysis of dirty pictures. In *J. of the Royal Stat. Soc. B*, pages 48(3):259–302, 1986.

[3] A. Broadhurst and R. Cipolla. A statistical consistency check for the space carving algorithm. In *Proc. ICCV*, 2001.

[4] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, pages 1039–1046, Sep 1999.

[5] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *Proc. ICCV*, Oct 2003.

[6] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH96*, 1996.

[7] M. E. Hansard and B. F. Buxton. Parametric view-synthesis. In *Proc. ECCV*, pages 191– 202, 2000.

[8] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons Inc, New York, 1975.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] M. Irani, T. Hassner, and P. Anandan. What does the scene look like from a scene point? In *Proc. ECCV*, 2002.

[11] R. Koch. 3D surface reconstruction from stereoscopic image sequences. In *Proc. ICCV*, pages 109–114, 1995.

[12] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.*, 20(3):127–150, 2001.

[13] T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. pages 825–832, 2004.

[14] D. Marr and T. Poggio. A computational theory of human stereo vision. *Phil. Trans. R. Soc. Lond. A*, 204:301–328, 1979.

[15] W. Matusik, C. Buehler, R. Raskar, L. McMillan, and S. Gortler. Image-based visual hulls. In *Proc. ACM SIGGRAPH*, pages 369–374, 2000.

[16] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH95*, 1995.

[17] D. M. Mount. ANN programming manual. Technical report, Dept. Comp. Sci., Univ. Maryland, 1998.

[18] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE PAMI*, 15(4):353–363, Apr 1993.

[19] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. CVPR*, pages 1067–1073, 1997.

[20] L. Wei. Deterministic texture analysis and synthesis using tree structured vector quantization. In *Proc. SIBGRAPH*, 1999.

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