Image Synthesis Considerations for Image Refocusing

Thomas Brow TEBROW@STANFORD.EDU

Abstract

Image refocusing from a single image is an underdetermined problem. Out-of-focus images of a scene do not capture high-frequency details that should appear in a focused image. In this paper, we describe a system for incorporating additional, focused images into the refocusing task in order to inform high-frequency detail. We use the additional inputs to determine a mapping from low-frequency to high-frequency features, and also to define a space of natural images to which we constrain our output. We find that the synthesized image accurately reproduces high-frequency textures from the scene when the focused inputs capture the scene from the same view, but achieves mixed results otherwise.

1. Introduction

Given an image in which some object appears out of focus, we would like to synthesize an image in which that object appears in focus. Since the image in a lens camera of an object at a given depth is the convolution of a pinhole image of that object by the point spread function (PSF) of the lens, a popular method for recovering the sharp image is deconvolution by the PSF. In deconvolution, blur is modeled by the equation

$$f * g = h, \qquad (1)$$

where f is the sharp image and g is the PSF. The output image is some solution for f given g and h.

However, the modulation transfer function (the magnitude of the Fourier transform of the PSF) of a typical photography lens falls to zero at a significantly lower frequency when the image is not in focus than when it is in focus [1]. Thus a band of spatial frequencies that should appear in a sharp image of the object is not present in the input image. Refocusing from a single image is therefore an underdetermined problem, and as a result, deconvolution algorithms are diverse and highly tailored to their respective applications.

We present a system that incorporates additional, focused images of the target object in order to inform the high-frequency detail of the synthesized image. Following [2] and [3], we use the additional inputs to define a space of natural focused images to which we constrain our synthesized image. As in those papers, we implement this constraint as a library of local patches appearing in the input images, requiring that every patch in the synthesized image resembles some patch in the library.

Our system further differs from deconvolution in that we do not generate candidates for the synthesized image by finding global solutions for (1). Instead, we find for each local patch in the unfocused image a set of patches from our focused patch library that are likely to resemble that patch when convolved by the PSF. This is analogous to the local approach toward super-resolution in [3], in which the set of high-resolution patches which downsample to a given low-resolution patch in the input image are considered. From this set of focused patches we derive a list of candidate values and corresponding probabilities for the center pixel, which can then be selected from according to the natural image constraint.

The primary advantage of our refocusing method over deconvolution methods is that it does not assume a particular strategy for recovering the lost high-frequency detail from the misfocused image, but instead lets a probabilistic mapping from low-frequency features to high-frequency features be specified by the additional inputs. Our method should thus retain greater generality across applications than any one deconvolution algorithm.

In section 2, we describe the components of our method in detail. In section 3, we compare the results produced by different variations of our system. In section 4, we evaluate the results of our best system. In section 5, we conclude the paper and suggest future work.

2. System

In the previous section, we described two probabilistic constraints on the synthesized image: our mapping of blurred to sharp neighborhoods, and our natural image constraint. Ideally, we would like to output the image which best satisfies the constraints across the entire image. In practice, we reduce the dimensionality of our search space considerably by instead considering pixels

individually. Each constraint is therefore represented as a probability distribution over values of a single pixel, and we seek to maximize the product of these probabilities for each pixel value assignment. We address the issue of inter-pixel dependencies and the order of pixel value assignments toward the end of this section.

Since we wish to avoid calculating both probabilities for each value for each pixel, we consider the constraints in order, evaluating the second constraint for only those values determined to be most probable by the first.

In formally defining these constraints, we use the symbols defined in Table 1.

Symbol	Meaning
U	Misfocused input image
V	Set of patches from focused input images
W	Synthesized image
G	Input PSF
$\|\|a - b\|\|^2$	Sum squared difference of a and b
$N(I, x, y)$	Neighborhood of pixel (x, y) in image I
$V[c]$	Subset of V which have center value c
$Blur(v)$	Blurred version of patch v

Table 1. *Table of symbols.*

2.1. The Image-Scene Constraint

In Section 1, we describe our method as finding a probabilistic mapping from low-frequency features to high-frequency features. Speaking more precisely, we maintain a mapping from misfocused image patches to focused patches that resemble them if blurred. This approach is an instance of *Image/Scene* training, where the observed *image* variable is a known function of the hidden *scene* variable. The function may be one that loses information, as in downsampling [3]. Because misfocus is such a function, the mapping is one-to-many, and can be represented probabilistically.

Note that our image-scene training task differs slightly from that of super-resolution. Whereas in downsampling the value of a low-resolution image patch is a precise function of the corresponding patch in the high-resolution scene, patches in a misfocused image receive light from pixels that fall outside of that patch in the focused scene. Our function from scene to image therefore has a probabilistic component itself.

We implement the image-scene mapping by a nearest-neighbor search in a library V of focused patches, where the query is our misfocused image patch. For each patch $v \in V$ we keep a blurred version $Blur(v)$, being the same rectangle extracted from the same input image convolved by G. The patch u in U maps to the focused patch v in V with a probability that is a function of the sum squared difference between u and $Blur(v)$. Taking the probability of the value c for the pixel (x, y) in W to be the probability of the most probable mapping from $N(U, x, y)$ to a patch in V with center pixel value c, we define

$$P_{image}(W_{x,y}) = \exp(-\lambda \min_{v \in V[W_{x,y}]} \left\| Blur(v) - N(U,x,y) \right\|^2).$$

To generate the list of best candidates for pixel (x, y) according to the image-scene constraint, we simply compare $N(U, x, y)$ against all blurred patches in V and select the center pixel values of the nearest matches.

2.2. The Scene Texture Constraint

Having generated a list of candidate values for each pixel in W, we choose among those values by constraining W to lie in the space of natural images. That is, we require that each patch in W resemble some patch in V. As with the image-scene constraint, we define the scene texture constraint by a probability distribution over values of a pixel,

$$P_{texture}(W_{x,y}) = \exp(-\lambda \min_{v \in V} \left\| v - N(W',x,y) \right\|^2),$$

where the image W' is the result of assigning $W_{x,y}$.

This neighborhood-based definition of the constraint implies inter-pixel dependencies in the synthesized image. If we use a symmetrical neighborhood – one in which for some pixel (a, b) in $N(W, x, y)$, (x, y) is in $N(W, a, b)$ – then circular dependencies exist [2], and we are forced at some point in synthesizing W to evaluate $P_{texture}$ for pixels whose neighborhoods contain unassigned values. Alternatively, we may choose a causal neighborhood, which only contains pixels that precede the center pixel in the order of assignment. Examples of causal neighborhoods under different assignment orders are given in Figure 1. Perhaps because of the smaller pixel-wise footprint, our results using causal neighborhoods were poor. We therefore used a symmetrical neighborhood, taking the most probable candidate values given by the image-scene constraint as initial values for W. In the next section, we consider which order of assignment best complements our choice of a symmetrical neighborhood.

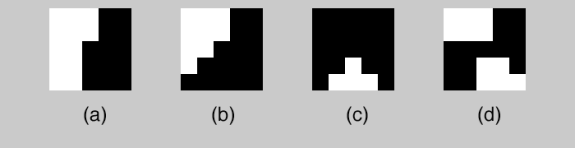


Figure 1. *Causal 5x5 neighborhoods under different assignment orders: the fixed neighborhood (a) of scanline-order assignment, and typical neighborhoods (b), (c), and (d) in confidence-order assignment. Black pixels are those not included in the neighborhood.*

2.3. Order of Assignment

Recent projects in image synthesis with neighborhood-based constraints have achieved results using symmetrical neighborhoods with a scanline or random assignment order, beginning from initial pixel value estimates and assigning each pixel more than once in multiple successive iterations [2, 6]. In [6], each pixel is weighted by confidence, such that it bears more strongly on neighborhood comparisons as it converges to its most probable value.

In multi-resolution texture synthesis – where the neighborhood used in synthesizing a given resolution level may extend into lower, already fully assigned levels – random order [5] and scanline order [4] assignment are successful even with a causal neighborhood. Because this synthesis begins with the lowest-resolution level, which is given, and proceeds upward, we might say that these algorithms assign pixels in confidence order. In conjunction with the causal neighborhood, this order ensures that the value of any pixel is determined only by pixels with equal or greater confidence.

We hypothesize that confidence-order assignment is also beneficial under a symmetrical neighborhood. However, our image space differs from the multi-resolution pyramid in that there is no obvious gradient from more confident to less confident pixels. The best notion of pixel confidence we are able to establish prior to assigning any pixels is the probability P_{image} of the most probable candidate value given by the image-scene constraint, normalized over all candidates. A map of this confidence metric over a sample input image is presented in Figure 2.

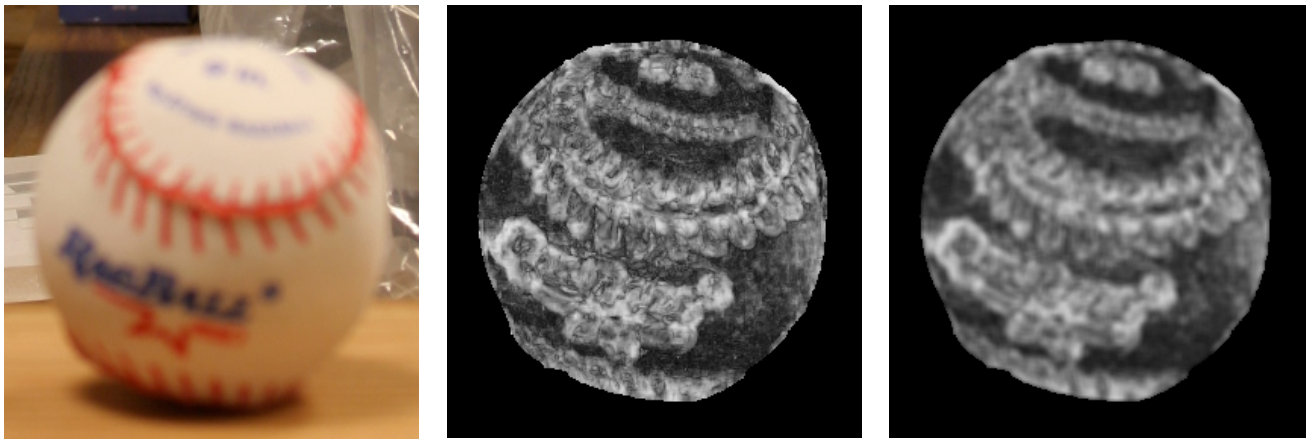
In general, the pixels of greatest confidence tend to lie in the areas of greatest contrast in the misfocused input image. These correspond to neighborhoods in the misfocused input image that resemble only very few neighborhoods in the blurred focused input images, and thus map very probably to a select few candidate values. Neighborhoods in large, uniformly colored regions resemble many patches in the blurred focused images, and thus map to many candidate values with roughly equal probability, resulting in low confidence.

Rather than use these raw pixel confidence values to determine the pixel assignment order, we consider that we would like to give precedence to pixels with high confidence values *in their neighborhoods* in W, since the value of a pixel is as much determined by the scene texture constraint as by the image-scene constraint. We therefore convolve the pixel confidence map by a 5 x 5 square to get a neighborhood confidence map, wherein the value of any pixel is the average pixel confidence across a 5 x 5 neighborhood around it. Sorting the list of pixels in W by this neighborhood confidence attribute yields our order of assignment.

3. Evaluating the Components

In order to determine the contribution of the components described in the previous section to the performance of the overall system, we compared the output of several variations of the system. For these evaluations we used the photograph shown in Figure 2(a) as our misfocused input image, and an in-focus photograph taken from the same view as our lone focused input image. Along with each input image we provide the program with a hand-made bitmask defining the boundary of the object to be refocused. Refocused values are synthesized only for the pixels within this boundary in the misfocused input, and our focused patch library is built only from the pixels within this boundary in the focused input.

In order to acquire an accurate PSF for the misfocused image, we photographed a simulated point light source placed at the same depth as the object, using identical focus and aperture settings. The point light source was simulated using a fluorescent desk lamp positioned behind a pinhole mask with a diffuser between. Since the ostensibly focused image of the light source was not in fact a point, but instead a few pixels in diameter, we took our actual PSF to be the function which when convolved by the focused image of the light source most closely resembled the misfocused image of the light source, constraining that PSF to be a disk with integer pixel diameter.



(a) (b) (c)

Figure 2. *Determining a confidence ordering over pixels: (a) the misfocused input image, (b) a map of pixel confidence over the synthesized image, and (c) a map of neighborhood confidence. In (b) and (c), brighter values correspond to higher confidence. The confidence maps suggest an assignment ordering that begins at regions of high contrast and radiates outward*

3.1. The Image-Scene Constraint

We first measured the performance of the image-scene constraint alone. In this variation of the system, the value of each pixel in the synthesized image is the most probable candidate for that pixel as defined by P_{image} in section 2.1. This yields the image shown in Figure 3(a).

Unsurprisingly, the performance of this variation in a given region corresponds to the confidence of the pixels in that region as described in section 2.3. In regions of high confidence, where a given neighborhood maps with high probability to a few select neighborhoods, the synthesized image very accurately reproduces the texture of the focused input image. These regions of high confidence generally correspond to regions of high contrast.

In more uniform regions, where each neighborhood in the misfocused input image may map with roughly uniform probability to a large number of patches in the focused input, the texture of the synthesized image appears noisy and does not resemble the real texture of the object.

3.2. The Scene Texture Constraint

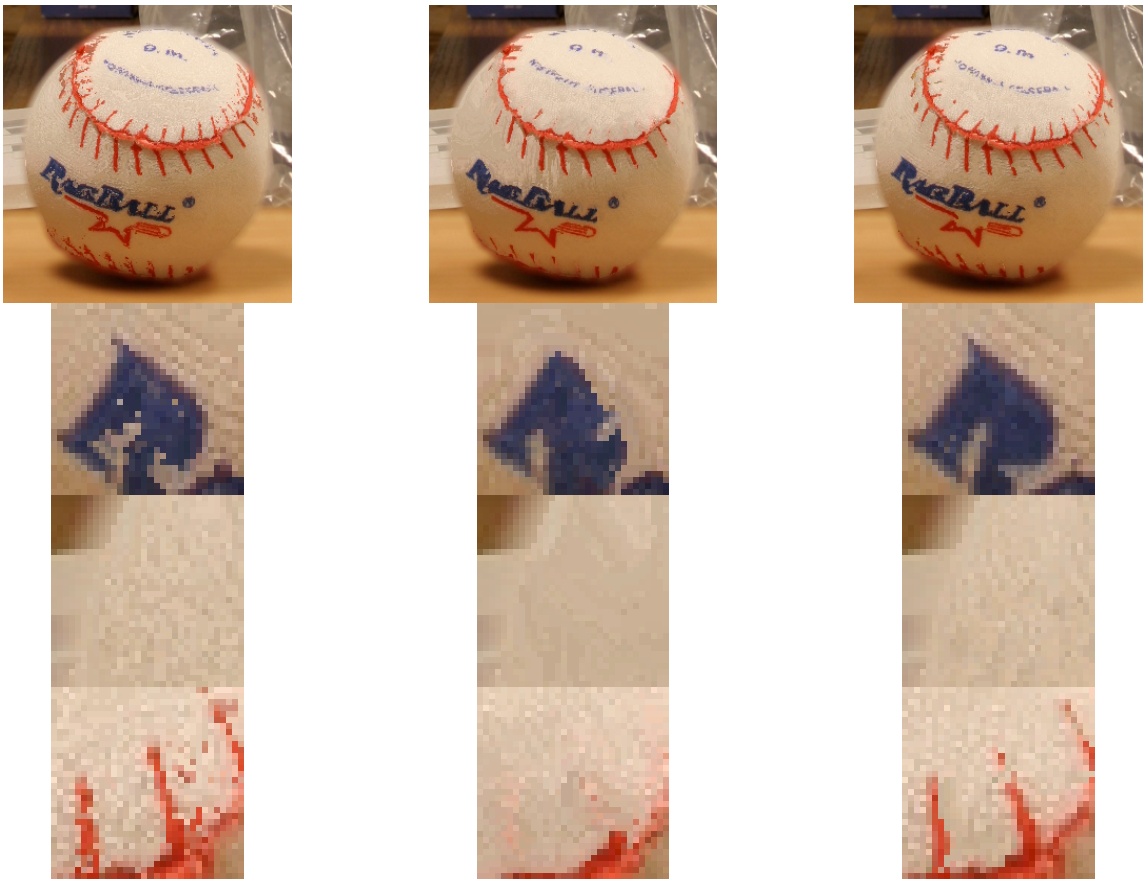
Figure 3(b) shows the results of incorporating the scene texture constraint into the variation of the previous subsection. In this variation, we calculated $P_{texture}$ for each of the top 20 candidates given by the image-scene constraint. We then selected the value v in the candidates that maximized

$$P_{image}(v)P_{texture}(v).$$

We performed this selection for each pixel of the synthesized image in scanline order, using a causal neighborhood in calculating $P_{texture}$.

The new constraint succeeds in eliminating the noisy texture that appeared in the low-contrast regions of the synthesized image when using the image-scene constraint only. However, the smooth, grain-free texture that it tends to render instead is also not correct. Furthermore, textures and colors appear to "bleed" out of their correct regions. In some cases, this causes small features on the target object to disappear entirely.

The bleeding effect may be accounted for by the relative weights of the constraints, which are determined by the tuning factor λ used in calculating P_{image} and $P_{texture}$. In a scanline ordering, excessive weight on the scene texture constraint causes textures currently established along the scanline to be propagated in the direction of the scan farther than should be allowed by the image-scene constraint. However, we do not experiment with different weights in this paper. Instead, in the next section, we consider how the bleeding effect may be mitigated by assigning pixels in order of confidence.



(a) (b) (c)

Figure 3. *Performance of variations of the system: (a) image-scene constraint only, (b) incorporating the scene texture constraint with scanline-order assignment, and (c) incorporating confidence-order assignment.*

(a) (b)

Figure 4. *Performance of the system using both constraints and confidence order assignment: (a) the misfocused input image, and (b) the synthesized image.*

Figure 5. *Performance of our best system given non-ideal inputs. Given the focused input image (a) with perturbed view of the target object, the system produces the synthesized image (b).*

3.3 Confidence-Order Assignment

Figure 3(c) shows the results of altering the variation of the previous subsection to assign pixels in confidence order as described in section 2.3.

The new assignment order largely mitigates the texture bleeding introduced by the previous variation. While some textures do encroach into other regions slightly, no features of the object are seen to disappear entirely as with scanline ordering. The textures generated by this variation also resemble the true texture of the object more closely than those generated by the previous variation.

Figure 4 shows another image refocused using this variation of the system.

4. Evaluating the System

We have demonstrated in the previous section that the variation of our system that incorporates both the image-scene constraint and the scene texture constraint and assigns the synthesized pixels in confidence order produces the most desirable results given an ideal focused input image. In this section, we evaluate the performance of that best system for non-ideal inputs. Specifically, whereas our focused input image in the previous section was taken from precisely the same view as the misfocused input image, we now consider the case of a focused input taken from a perturbed view. The results of using such an input with the misfocused images of the previous section are shown in Figure 5.

The performance of our system under these conditions clearly falls short of the performance on ideal inputs as some regions of the synthesized regions take on incorrect textures while others take on incorrect overall color as well. As a result, some features of the object disappear. These results suggest that our method for mapping neighborhoods in the misfocused image to neighborhoods in the blurred focused image is not robust to changes in the viewpoint of the focused image. This may result in a poor selection of candidate values for the synthesized pixels, such that no good image exists in the space of candidate images, and the performance of the system as a whole is thusly limited.

5. Conclusions and Future Work

We have stated it as the goal of this paper to demonstrate a method for image refocusing which is both more generally applicable than deconvolution algorithms and better able to take advantage of additional information about high-frequency detail.

Our method indeed achieves the latter in the broadest sense, as we have demonstrated the ability of our system to replicate the high-frequency details of focused input images convincingly in some regions of synthesized images. Traditional deconvolution algorithms do not incorporate focused images at all. We have also seen, however, that the system is not able to use focused images effectively that are not taken from the same view as the misfocused image. Thus the utility of our refocusing method is currently limited to situations where we already have a suitable focused image from the desired view, which makes it less applicable than most deconvolution algorithms.

We stand to improve the performance and generality of the method by exploring various parameters of the method that this paper does not, including neighborhood size, our choice of pixel confidence metric, and the relative weights of our constraints. However, the computational cost of running our system makes it difficult to experiment with all of the parameters we would like to. Our current best method takes 5 to 10 hours to synthesize a 300 x 300 image using 20 candidate values per pixel on a vine box. We therefore propose future work that aims at increasing the efficiency of our system.

5.1. Intelligent Candidate Selection

In the experiments presented in this paper, we considered only the 20 most probable candidates generated by the image-scene constraint as possible values for the pixel. This is a very small fraction of the total number of candidates, which may be as great as the number of pixels in the focused input images. Moreover, when our confidence in the pixel is low, values outside of the top 20 may have only marginally lower probabilities than those that make the cut. While it is necessary to restrict the number of candidates we consider to keep the system computationally feasible, there may be a more intelligent approach to selecting those candidates.

Such an approach might address the fact that our current system considers each value in RGB space as a separate candidate. Thus if we have a candidate value (0, 0, 1) with probability 0.3, a candidate (0, 1, 0) with probability 0.3, and a candidate (255, 255, 255) with probability 0.4, then "white" is regarded as the most probable candidate, whereas the true value is probably very close to "black".

A more sensible procedure for candidate selection might, for example, perform *k*-means clustering in RGB space on the entire set of candidate values, with each value weighted by its probability. One could set $k = 20$ and take the final means as candidate values.

5.2. Accelerated Neighborhood Search

The run time of our system is dominated by neighborhood searches, which are nearest-neighbor searches in 75 dimensions (5 x 5 patches in 3 channels). Our current implementation is exhaustive linear search through the patch library, which runs in time O(N) on a library of N patches. However, approximate nearest-neighbor algorithms exist which after a reasonable pre-processing step can perform this search in time O(log N), and which have performed satisfactorily in texture synthesis [4] and image synthesis [2] .

The obstacle to implementing our current system on top of these algorithms during experimentation was that we required the ability to locate the nearest neighbor in an arbitrary subset of the 75 dimensions, in order to support arbitrarily shaped causal neighborhoods. Now that we have identified a best system that uses symmetrical neighborhoods, any ANN or vector quantization algorithm may be used for neighborhood search.

Acknowledgements

I would like to thank Marc Levoy for providing and providing assistance with photography equipment and props for this project, and Hendrik Lensch for his comments and suggestions on my initial proposal.

References

[1] W. T. Cathey and E. R. Dowski, "New paradigm for imaging systems," *Appl. Opt.* **41,** 6080-6092.

[2] A. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-Based Rendering Using Image-Based Priors,"

[3] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning Low-Level Vision," *IJCV* **40,** 25-47.

[4] L. Wei and M. Levoy, "Fast Texture Synthesis using Tree-structured Vector Quantization," *Proc. SIGGRAPH 2000*.

[5] L. Wei and M. Levoy, "Texture Synthesis over Arbitrary Manifold Surfaces," *Proc. SIGGRAPH 2001*.

[6] Y. Wexler, E. Shechtman, and M. Irani, "Space-Time Video Completion," *Computer Vision and Pattern Recognition* **1,** 120-127.