Dense View Interpolation on Mobile Devices using Focal Stacks

Parikshit Sakurikar and P. J. Narayanan
Center for Visual Information Technology
International Institute of Information Technology - Hyderabad, India
{parikshit.sakurikar@research.,pjn@}iiit.ac.in

Abstract

Light field rendering is a widely used technique to generate novel views of a scene from novel viewpoints. Interpolative methods for light field rendering require a dense description of the scene in the form of closely spaced images. In this work, we present a simple method for dense view interpolation over general static scenes, using commonly available mobile devices. We capture an approximate focal stack of the scene from adjacent camera locations and interpolate intermediate images by shifting each focal region according to appropriate disparities. We do not rely on focus distance control to capture focal stacks and describe an automatic method of estimating the focal textures and the blur and disparity parameters required for view interpolation.

1. Introduction

We present a dense view interpolation method for mobile devices in which a few images of the scene are captured in the form of focal stacks, and novel views of the scene are generated by interpolating the focal textures generated from these focal stacks. We use a model in which the scene is assumed to consist of a set of focal planes. From the captured focal stack at each viewpoint, an all-in-focus representation of the view is generated by merging the focused regions of each focal slice into one image. The focused regions are then shifted by automatically estimated disparities to generate novel views from novel viewpoints.

Synthesis of novel views for a given scene relies on either geometric estimation of the scene elements or image-based rendering [5, 9]. While geometric methods rely on accurate estimation of the depth of scene elements, image based methods generate novel views by interpolating a set of captured images. Light field rendering is an image based method which uses dense sampling of light rays for novel view synthesis. For aliasing free light field rendering, a very high sampling density of camera locations is necessary, as described by Chai et al. [2].

In order to achieve a dense sampling of scene light rays, it is useful to capture images at sparse camera locations and interpolate a dense set of intermediate images between them. We use an iterative method to estimate the focal textures at each camera location and interpolate these focal textures based on appropriate disparity estimates to generate the intermediate images. Dense view interpolation for a scene consisting of two manually-identified depth layers was presented by Kubota et al. [6]. Levin et al. [8] enhanced this idea by capturing a focal stack of the scene and generating novel views of the scene from different viewpoints within the aperture area by geometrically shifting each focal slice with its appropriate disparity.

In our case, we model the scene as consisting of multiple focal regions, but do not need to capture an exhaustive focal stack. We also do not need to identify the focal regions manually. We use a mobile device to capture an adaptive focal stack of the scene from adjacent camera positions and adaptively estimate the disparity and blur parameters required for accurate view interpolation between the camera positions. We use mobile devices as they are highly accessible and they tightly integrate optical lenses, electronic sensors and versatile computing. Such a combination is ideal for computational photography which often requires adaptive optical sensing and embedded computing [1].

2. Imaging and Interpolation Model

We assume that the scene consists of objects at n different depth-of-field regions and we capture focal stacks from several adjacent camera locations. The imaging model treats each captured image as a combination of ideal *focal textures* located at n focal regions of the scene.

A focal stack g^i is a set of differently focused images captured using a camera with a finite aperture. The scene consists of n focal textures f^i_j at each of the n focal planes captured in the focal stack. These focal textures at a given camera location C^i can be understood as

$$f^i_j(x,y) = \left\{ \begin{array}{cc} f^i(x,y) & \text{if } z^i(x,y) = j \\ 0 & \text{otherwise} \end{array} \right\} \quad (1)$$

where f_j^i is the j^{th} focal texture, f^i the ideal all-in-focus image at the camera location C^i, and $z^i(x,y) \in [1,n]$ is the index of the focal slice at which the pixel (x,y) is maximally in focus. The true f_j^i textures are unknown. The focal stack g^i can be modeled as a sum of these textures convolved with appropriate blur kernels as

$$g_k^i = \sum_{j=1}^{n} h_{kj} * f_j^i, \quad k \in [1,n] \ , \tag{2}$$

where h_{kj} represents the blur kernel (usually a Gaussian of blur radius R_{kj}) between the focal planes k and j with h_{kk} being a delta function. Equation 2 is an extension of [6] and approximates the focal stack using the focal textures and blur kernels. This linear imaging model may not be correct at depth discontinuities, but does not produce significant interpolation artifacts. With known blur kernels, Equation 2 consists of n equations in n unknown f_j^i textures.

Once the f_j^i textures are estimated, the all-in-focus image f^i can be evaluated as

$$f^i = f_1^i + f_2^i + f_3^i + \ldots + f_n^i \ . \tag{3}$$

Our objective is to compute the all-in-focus image at an arbitrary intermediate location, between two focal stacks from locations C^i and C^{i+1}. We parameterize the location using the fraction α of the distance between them, with a value of 0 at C^i and a value of 1 at C^{i+1}.

The focal textures f_j^i and f_j^{i+1} can be interpolated to generate the intermediate image as shown in Figure 1. The f_j^i textures are shifted forwards by αd_j and the f_j^{i+1} textures are shifted backwards by $-(1-\alpha)d_j$, where d_j is the disparity of the j^{th} focal texture between camera locations C^i and C^{i+1}. The disparity d_j can be either horizontal or vertical based on the change in position from C^i to C^{i+1}. The interpolated forward and backward all-in-focus images for a horizontal shift can be generated similar to Equation 3 by first shifting the f_j^i textures and then adding them to generate the all-in-focus intermediate image:

$$f'(x,y;\alpha) = \sum_{j=1}^{n} f_j^i(x - \alpha d_j, y) \tag{4}$$

$$f''(x,y;\alpha) = \sum_{j=1}^{n} f_j^{i+1}(x - (\alpha - 1)d_j, y) \tag{5}$$

A similar expression can be derived for shifts between vertically separated camera positions.

The all-in-focus image f_α at the intermediate location α can be approximated using a weighted average of f' and f'' [6] as

$$f_\alpha(x,y) = (1-\alpha)f'(x,y;\alpha) + \alpha f''(x,y;\alpha). \tag{6}$$

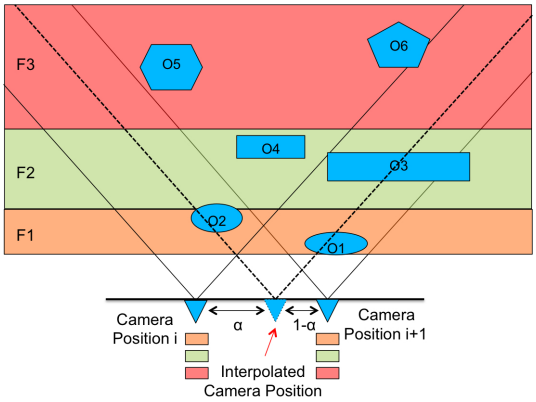

Figure 1. All-in-focus image at α is interpolated from images at C^i and C^{i+1}. Scene objects denoted by O_1 to O_6 and focal zones denoted by F_1 to F_3. The focal slices are refined from the captured focal stack such that $F_1 \cup F_2 \cup \ldots \cup F_n$ covers the entire desired range and $F_1 \cap F_2 \cap \ldots \cap F_n$ is negligible.

3. Capturing the Focal Stack

The focal stack g^i is a set of differently focused images captured at camera location C^i. A focal stack can be used for generating all-in-focus views by depth-of-field extension, for post-capture refocusing over the scene, for generation of arbitrarily refocused images and also for view interpolation by shifting the focal regions by correct amounts. Ideally, a focal stack is captured by controlled motion of the camera lens relative to the camera sensor by predefined distances. The distance of lens motion in each step is decided such that a minimal number of images are captured while ensuring that the entire depth variation of the visible scene is covered, as described in [11]. It is possible to implement such a capture mechanism on certain DSLR cameras which enable live control of focal length and aperture settings, and on certain Linux based mobile phones which allow for programmatically controlling the lens focus distance [10]. This control is however not available on a large number of mobile devices and cameras in which there is no explicit programmatic control of the lens focus distance like most Android based smartphones. The Android SDK provides a function call to only retrieve the focus distance of the camera but the function returns unreliable and incorrect results on several devices.

We present a simple method to capture a focal stack on devices that do not possess explicit focus distance control. Most cameras and smartphones are equipped with auto-focus and touch based manual-focus features which can fix the focus of the camera based on the user's desired focus location. We make use of this focus control mechanism in order to capture a set of images with different focus settings and emulate a focal stack from these captured images.

In order to capture images focused at different locations in the scene, we divide the visible scene into sixteen dif-

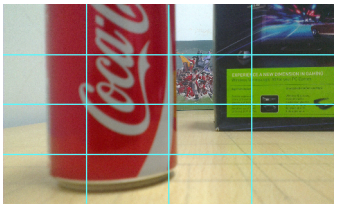

Figure 2. The focal stack is captured using 16 focus regions.

ferent rectangular regions as shown in Figure 2. Images are captured by sequentially setting the focus area to each of these regions. After capturing these images, the image slices that might be redundant (due to different rectangular focus regions mapping to the same physical focus distance) are eliminated by measuring absolute image differences between the captured slices. As the redundant images are similar in both the focused and defocused regions, analyzing pixel differences works well even though there might be small misalignment between the captured images due to camera shake. The focal stack is thus emulated as a set of a few images having different focus distances, without any explicit order between the images and without explicit knowledge of the focus distance itself. Thus, interpolation of the focal textures based on geometric disparities similar to [8] is not directly possible. However, with explicitly estimated disparities for each focal slice between adjacent camera locations, it becomes possible to interpolate intermediate views.

4. Focal Textures and View Interpolation

To handle scenes with unknown number of focal regions, we need to estimate the number of focal regions in the scene. We assume that the number of images remaining after eliminating redundant focal slices corresponds to the different visible focal regions in the scene. For view interpolation, we need to estimate the textures f_j^i, the disparities d_j, and the blur radii R_{kj} from the available focal stack g^i.

Figure 3. The focal stack at a certain C^i consisting of three focal slices, along with the sharpness index map built over the stack.

We capture a focal stack g^i at different camera locations C^i such that there is a small horizontal or vertical separation between any two adjacent camera locations. Each focal stack g^i at C^i is explicitly registered to the first focal slice g_1^i. This is done in order to eliminate magnification errors due to focus/defocus and to account for camera motion during capture. The time taken for the described focal stack capture mechanism is relatively large as many images are being captured with a change in focal settings after each capture. This can possibly lead to misalignment in the focal slices. In order to register the focal stack, we use the Enhanced Correlation Coefficient [3] algorithm, which uses a modified correlation coefficient model to accurately register images suffering from photometric distortions. Registering the focal stack ensures per-pixel alignment of the stack.

4.1. Sharpness Index Map

In order to estimate the blur and disparity parameters required for view interpolation, we make use of a sharpness index map built over the focal stack. In the index map, each pixel across the stack is labeled to the focal texture at which the weight w_k is maximized, where

$$w_k(x, y) = \nu(P_k(x, y, d)) \tag{7}$$

represents the variance of a patch of size d centered at the pixel in the g_k^i image [11]. Thus, the index map for any focal stack labels each pixel of the registered stack to the image where it appears maximally sharp (in-focus). Figure 3 shows the index map built from the focal stack. The index map so generated can now be used to automatically build rectangular focal templates required for further parameter estimation.

4.2. Estimating Disparities

We can identify the disparity d_j between adjacent camera locations for every focal plane j using the index map. We use template matching based on FFT correlation, using automated selection of templates. For each focal image g_k^i, we use the index map to locate a set of rectangles in which all pixels are labeled to the same focal region, similar to connected component analysis over pixels. We collect these rectangles in a list sorted by size. We select the largest located rectangle as the focal template and match it in the same focal image of the next camera position g_k^{i+1} to estimate the disparity d_k between the two camera locations for that layer. The focal templates extracted from the index map are shown in Figure 4.

The disparity estimation method may fail for uniform textured regions if pixels do not register correctly to their true focal region. Also, if the template is large, then it may be partially occluded in the adjacent image because of change in camera position. To solve this, we discard template matches with a high vertical disparity when the camera movement between C^i and C^{i+1} is horizontal and vice versa, or if the disparity for a focal region is abnormally

Figure 4. The focal templates for the three focal regions extracted from the sharpness index map by generating a list of uniform index rectangles sorted by size.

large or small compared to neighboring focal disparities, and use the next largest rectangular template.

4.3. Blur kernel estimation

The blur kernels h_{kj} can also be estimated automatically once we have extracted focal templates for each focal region. They are estimated by sequentially blurring the template with incremental blur radii until the best match is found. The blur radii are estimated for every R_{kj} pair but we assume that the defocus blur between the focal layers k and j would be constant for the two focal slices i.e. $R_{kj} = R_{jk}$. Thus for a scene consisting of three focal regions, three blur parameters of R_{12}, R_{13} and R_{23} are estimated. The defocus blur over the chosen templates is shown in Figure 5. Small errors in blur estimation have been empirically shown to cause little or no interpolation artifacts [6].

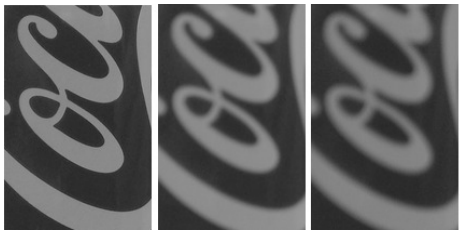

Figure 5. The first focal template extracted from the f_1^i, f_2^i and f_3^i textures. The defocus blur between focal regions is estimated using these templates.

4.4. Solving for Focal Textures

Once the focal stack g^i is aligned due to explicit registration and the d_j and h_{kj} parameters are estimated, we solve Equation 2 in the frequency domain to estimate the f_j^i textures as follows

$$G_k^i = \sum_{j=1}^{n} H_{kj} F_j^i, \quad k \in [1, n] \tag{8}$$

Figure 6. The focal stack at a certain C^i along with the all-in-focus image.

F, G and H are the Fourier transforms of the f textures, the images g and the kernels h respectively, with $H_{ii} = 1$. Equation 8 gives a system of n equations in n unknowns, where each variable is a matrix in the frequency domain. In [6], the acquired equation is a system of two equations in two variables and interpolated views are generated using linear filtering of the input images without explicitly identifying the F images, but filter corrections are applied to eliminate artifacts caused by H_{kj} filter divergence at DC.

We require a general solution for Equation 8 with n focal layers. We iteratively solve for the F textures in the frequency domain. We use an iterative method similar to [7] but solve for the focal textures in the frequency domain as expensive convolution operations in each linear equation are replaced by per-element matrix multiplication. After k iterations:

$$\begin{aligned}
F_1^{i,k} &= G_1^i - [H_{12} F_2^{i,k-1} + H_{13} F_3^{i,k-1} \\
&\quad + \ldots + H_{1n} F_n^{i,k-1}] \\
F_2^{i,k} &= G_2^i - [H_{21} F_1^{i,k} + H_{23} F_3^{i,k-1} \\
&\quad + \ldots + H_{2n} F_n^{i(k-1)}] \\
&\quad \ldots \\
F_n^{i,k} &= G_n^i - [H_{n1} F_1^{i,k} + H_{n2} F_2^{i,k} \\
&\quad + \ldots + H_{n(n-1)} F_{n-1}^{i,k}]
\end{aligned} \tag{9}$$

where $F_j^{i,k}$ represents the F_j^i texture after the k^{th} iteration. We apply the Gauss-Seidel iterative method to a system of linear equations in non-square matrices in the frequency domain [4]. Such a solution assumes a prior on the F textures. We use the sharpness index map to segment each g_j^i image into a prior f_j^i texture. Thus, all the pixels labeled to certain index j are picked from the focal stack image g_j^i to generate $f_j^{i,0}$. This estimates all the focal textures as a collection of best focused pixels at that focal plane. We solve Equation 9 using these priors and compute the inverse Fourier transform of the F matrices to estimate the f_j^i textures at each camera location. This solution of f_j^i textures does not require any explicit control over the blur kernels as in [6] and

Figure 7. Horizontally interpolated all-in-focus images at $\alpha = 0, 0.2, 0.4, 0.6, 0.8, 1.0$. Expanded views of the images are shown below each image. Different focal textures being shifted by their appropriate disparity is visible.

is scalable to many focal layers. The overall procedure involving the estimation of focal textures and interpolation is described in the following algorithm

Algorithm 1 PROCEDURE

1: Capture an approximate focal stack at each C^i. Evaluate number of visible focal layers n.
2: Register each image in g^i to g^i_1.
3: Estimate the sharpness index map over the focal stack.
4: Estimate the disparity d_j between adjacent stacks for each layer j.
5: Estimate blur radii R_{kj}.
6: Solve Equation 2 for f^i_j.
7: Interpolate to get the α images.

4.5. View Interpolation

Once the f^i_j textures are computed, we can interpolate novel α views between any two camera locations using equations 4 and 5. The all-in-focus image can be estimated using Equation 3. As an example consider four clockwise camera locations forming a square C^{1-4}. An intermediate image at α and β (horizontal and vertical shifts) can be estimated by first interpolating the (f^1_j,f^2_j) and (f^3_j,f^4_j) along the horizontal direction using equations 4,5 and then correspondingly along the vertical direction. Thus the interpolation is composed of three sets of focal texture shifts and summation. The overall interpolation can be written as:

$$\begin{aligned} f_{\alpha\beta} =& (1-\alpha)(1-\beta)f^1 + \alpha(1-\beta)f^2 \\ &+ (1-\alpha)\beta f^3 \qquad + \alpha\beta f^4 \end{aligned} \tag{10}$$

where

$$f^i = \sum_{j=1}^{n} f^i_j(x - \alpha d^x_j, y - \beta d^y_j) \quad i \in [1,4] \tag{11}$$

The steps $1-6$ in the described Algorithm 1 can be computed as pre-processing steps in the background once the focal stacks are captured and the interpolation in step 7 requires lesser computational resources and can be applied actively based on the user's α, β input. Generating a set of dense interpolated views of the scene can also enable novel view generation using light field rendering from novel viewpoints that are not explicitly on the camera plane.

5. Experiments

We experiment with an HTC One X mobile device which is equipped with a quad-core 1.5GHz Nvidia Tegra 3 processor having an inbuilt ULP GeForce GPU. We use the OpenCV for Tegra 2.4.5 library with the Tegra GPU enabled for all OpenCV primitive image processing and core function calls. This device uses an 8MP camera which has a maximum aperture of f/2. The camera can capture 5 shots per second in burst mode without any intermediate autofocus calls. For capturing the focal stack however, there is a need to adjust the focus after each image and it thus takes about 0.8 to 0.9 seconds to capture an image. Our focal stack is a set of 16 images where each 1280x760 image is focused on a different rectangular region of the visible scene. We find a 4x4 block to be the ideal size for the focus regions as it captures most of the elements in the scene and also has a reasonable capture time. 3x3 or 5x5 blocks

would lead to objects being missed from focus and large capture times respectively. For two slices i and j, j is classified as redundant if the sum of absolute pixel differences of i and j is less than 10% of the maximum pixel difference across the stack for i. Pixel differences are estimated at half the image resolution for computational efficiency. The redundant image removal can also be pipelined with camera focus adjustment for each shot during capture.

We capture a focal stack at four camera locations lying on the edges of a $1.5cm$ square at an overall capture time of about one minute. The focal stacks are then registered and the pixel difference approach is reapplied to eliminate any redundant image that might not have been identified without per-pixel alignment. This step helps in case of significant camera shake during capture. The index map is built at one camera location and the focal templates for each focal texture are isolated.

The disparity and blur parameters are estimated using the focal templates. Since the relative distance between the camera positions is low, we can reuse the same templates to identify disparities between all camera locations. Also, the blur kernels are dependent on the distance of the scene objects from the camera and thus for planar movement of the camera along the square, the blur kernels are constant at all four camera locations. Estimating the index map, the blur radii and the focal templates is thus done once and takes about 2 minutes. Image registration, disparity estimation and focal texture estimation takes about $10, 15$ and 40 seconds respectively per camera location. The focal textures at each camera location are estimated by processing each color channel independently using Equation 9. The overall capture and processing time is about 2.5 to 3 minutes per camera location.

The interpolation of novel views is performed in near-real-time based on user input by shifting the focal textures by the appropriate disparities and merging them to generate the intermediate view. The processing involving the generation of focal textures can be fully computed on the device. However, considering the time taken for processing and the fact that the battery usage for a mobile device should be as low as possible, this approach can also be extended to a cloud based framework, where the device captures, registers and removes the redundancies from the focal stack, and the resulting images are uploaded to a cloud service which processes and feeds the focal textures back to the device for real-time view interpolation. The results of all-in-focus view generation and interpolation are shown in Figures 6,7.

6. Conclusion & Future Work

We have demonstrated a method of using mobile devices to capture focal stacks over a general scene and apply interpolative synthesis over the estimated focal textures to generate novel views. The method can enable a general user to capture a scene with small control of camera positions and view it later in a light field rendering framework from novel interpolated viewpoints. As mobile devices become equipped with larger aperture cameras, this method would enable more accurate view interpolation as each focal layer would correspond to a narrow depth-of-field and the method itself is scalable to many focal layers. We are working on a quick focus estimation method which would enable capturing a focal stack in a significantly shorter amount of time, by analyzing the focal information in the scene before capture.

References

[1] A. Adams, E. Talvala, S. Park, D. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico, H. Lensch, W. Matusik, K. Pulli, M. Horowitz, and M. Levoy. The frankencamera: An experimental platform for computational photography. SIGGRAPH 2010, page 29. ACM.

[2] J. X. Chai, X. Tong, S. C. Chan, and H. Y. Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 307–318, 2000.

[3] G. Evangelidis and E. Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1858–1865, 2008.

[4] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

[5] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 43–54. ACM, 1996.

[6] A. Kubota, K. Aizawa, and T. Chen. Reconstructing dense light field from array of multifocus images for novel view synthesis. *IEEE Transactions on Image Processing*, 16(1):269–279, 2007.

[7] A. Kubota, K. Takahashi, K. Aizawa, and T. Chen. All-focused light field rendering. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, EGSR'04, pages 235–242, 2004.

[8] A. Levin and F. Durand. Linear view synthesis using a dimensionality gap light field prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1831–1838, June 2010.

[9] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 31–42, 1996.

[10] D. Vaquero, N. Gelfand, M. Tico, K. Pulli, and M. Turk. Generalized autofocus. In *IEEE Workshop on Applications of Computer Vision (WACV'11)*, January 2011.

[11] C. Zhou, D. Miau, and S. K. Nayar. Focal sweep camera for space-time refocusing. *Technical Report, Department of Computer Science, Columbia University*, 2012.