Artificial Refocusing of High Resolution SLR Images From Microsoft Kinect Depth Maps

Christian Castellanos
Stanford University
ccastell@stanford.edu

Andrew Nguyen
Stanford University
aknguyen@stanford.edu

Abstract

We present a novel system for allowing the digital refocusing of high resolution images (captured with a single-lens reflex (SLR) camera) through the fusion of low resolution range data acquired via a Microsoft Kinect. We rely on the IR camera array inside of the Microsoft Kinect v2 for constructing a depth map. The pixel intensities of the output depth map corresponds to the relative depths of points in the image. The depth map is then used to segment the image into a user specified number of focus regions. The user can select which region they would like to focus on in the higher resolution image, and a simulated shallow depth of field effect is applied to the remaining regions in the image.

1. Introduction

1.1. Motivation and Background

A shallow depth of field in images is a powerful artistic tool for hobbyist and professional photographers alike. The ability to apply post-processing refocusing on specific depth planes in the image is especially lucrative, and few conventional methods exist for doing so with high resolution images. Light field cameras, such as the Lytro Illum, can compute depth information and provide on-the-fly refocusing techniques in the post-processing stage, but image quality and resolution suffers with such cameras.

Recomputing where the focus should be in an all-in-focus image, and applying a depth of field effect to the remaining areas of the image, can be done feasibly with the use of range data or a depth map. Some solutions for computing a depth map include Time-of-flight (ToF) cameras and LiDAR (Light Detection and Ranging). These solutions estimate the depth of a scene by illuminating it with light, and the camera measures the amount of time between initial pulses and the amount of time it takes for the camera's sensor to detect reflected pulses. The varying time of flights across a scene are used to build a depth map. Depth maps provide great potential for the accurate refocusing of scenes in the post-processing stage.

1.2. Paper Description

In this paper, we demonstrate how the noisy, error-prone, and low resolution depth information from a Microsoft Kinect v2 can be fused with a high resolution photograph of the same scene taken by an SLR camera to allow post-processing refocusing. The refocusing applies a shallow depth of field effect to different regions in the image as specified by the user.

We develop a method for the rectification of the noisy and corrupted depth data in conjunction with segmentation into user-defined focus planes. The depth map is thus upsampled and aligned with the SLR image such that the refocusing operation can be applied to the higher resolution image. The segmented data is used to compute masks that allow the refocused image to incorporate the mixing of focused foreground pixels and blurred background pixels.

1.3. Related Work

Work done by F. Moreno-Nogeur, P. Belhumeur, and S. K. Nayar in [3] allows the active refocusing of images via segmentation into depth maps, using a series of illuminated dots (structured illumination). A series of dots are projected onto a scene via a projector and an image is captured. The dots are used to estimate the varying depths in the scene and the scene is eventually separated into a background and foreground with the dots removed from the scene. The image can then be refocused into two regions: foreground and background. Their results show very realistic refocusing. The method presented in the paper only allows refocusing on two regions; additionally, it does not work as well in well, artificially lit scenes without taking an additional photograph. The use of alpha mattes for more realistic refocusing inspired us to do so as well.

Realistic refocusing with range data was accomplished by Benjamin Huhle in [1] but with a much different ap-

proach. Range data is acquired from a scene and is used to apply a continuous refocus, but also relies on taking multiple LDR photos at different exposures to compute a HDR image. His project also incorporates simulation of different apertures and exposure settings, a much more computationally expensive task.

2. Overview of Method

Figure 1: Proposed Image Processing Pipeline

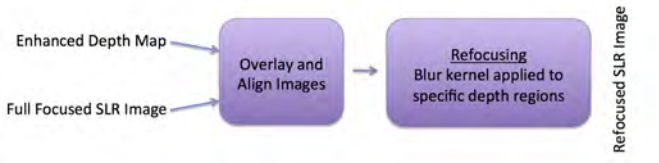


Figure 2: Refocusing Pipeline

Here we present a general overview of the proposed imaging pipeline stages necessary for our implementation of post-processing refocusing. Fig. 1 and 2 refer to the pipeline stages.

Image and Depth Acquisition: We set up our scene with respect to the Microsoft Kinect v2's dpeth sensing limitations. We take a photo of the scene with the Kinect's IR cameras to acquire a depth map, and to the best of our ability, take a photo of the scene with an SLR camera in such a way that image aligns with the depth map.

Rectification: The output of the Kinect depth sensor suffers from occlusions and invalid depths, particularly at edges. A hole-filling algorithm is used to better estimate the relative depths of these pixels.

Segmentation: Over-specified K-means is used to segment the depth map into more regions than necessary. The user then selects which clusters belong to the region he or she would like to have in distinct focus regions and relabels the clusters accordingly.

Upsampling and Alignment: The depth map must be upsampled to match the resolution of the higher resolution SLR image. The depth map must also be aligned properly with the SLR image as best possible given aberrations resulting from the Microsoft Kinect. Automatic alignment was outside of the scope of the project, so it must be done manually in an application such as Photoshop or GIMP.

Matting: The user specifies which region from the upsampled image should be in focus. That region is designated the foreground, while the other regions are the background. Using the upsampled depth map, an alpha matte is computed on the high resolution SLR image. The alpha matte assesses the probability that a given pixel is a member of the foreground.

Refocusing SLR Image:The alpha matte and its complement are used to compute the final refocused image. The shallow depth of field effect is simulated via an airy disk kernel with a user specified radius. This allows for artistic shallow DoF effects.

3. Image Acquisition

First a scene is setup with respect to the maximum and minimum ranges for accurate depth sensing. The Kinect v2 is limited to sensing depths between .5 and 4.5 meters, as well as odd field-of-view constraints. The scene can be dimly or well lit; however, sunlight tends to interfere with the IR depth camera.

Using the Microsoft Kinect v2 with a computer requires the special Kinect for Windows Adapter. Windows is not required, as an API for using the Kinect with Processing (an open source image processing platform available for most platforms) was developed by Daniel Shiffman, OpenKinect. Using OpenKinect with specific depth API calls allowed us to save depth map data from the Kinect's IR camera. Depth maps acquired from the Microsoft Kinect suffer from occlusions and inaccurate readings at the edges of the scene and around edges of objects; these inaccurate or unknown readings are rendered as black pixels. The resolution of the depth camera output is 512x424, or roughly 7x7 pixels per visual degree. An example of raw depth data with the depth holes can be seen in Figure 3a.

Without modifying the structure of the scene and ensuring that the lighting was the same, we used a 18MP SLR Camera to take a shot of the scene, using the distorted depth map acquired from the Kinect (see Section 11.2, **Microsoft Kinect Camera**). The SLR photo must be taken all-in-focus for the refocusing and shallow DoF effect to be digitally applied in the proposed processing pipeline.

(a) Raw Kinect Depth Map Data (b) Hole-Filled Depth Map

Figure 3: Raw depth map applied with the hole-filling algorithm to remove unknown pixel holes.

4. Hole Filling

The depth map output by the Kinect's IR camera suffers because of occlusions and estimations of depth on the edge of the scene. These pixels are rendered invalid in the depth map, and have intensity value 0 (black). In order to proceed with the segmentation step, the pixels resulting from occlusion must be given valid (and presumably accurate) intensity values corresponding to their depth. We use a simple-yet-effective hole-filling algorithm for assigning these pixels proposed in [2] to do so.

For every invalid ($\mathbf{0}$) pixel, $\mathbf{p}$, inside the image, the algorithm examines an $\mathbf{NxN}$ region around the pixel. It finds the largest intensity value, $\mathbf{i}$, inside the window and sets the value $\mathbf{p} = \mathbf{i}$. The user chooses the size of the filter window, $\mathbf{N}$. If no nonzero intensity values are found, the invalid pixel is unchanged. The hole filling algorithm functions very well on occlusions, but suffers when depth maps are taken that do not respect the range limitations of the Kinect. In these depth maps, the hole-filling algorithm would have trouble filling in large regions of black pixels. Figure 3b shows the results of the hole filling algorithm.

5. Depth Map Segmentation

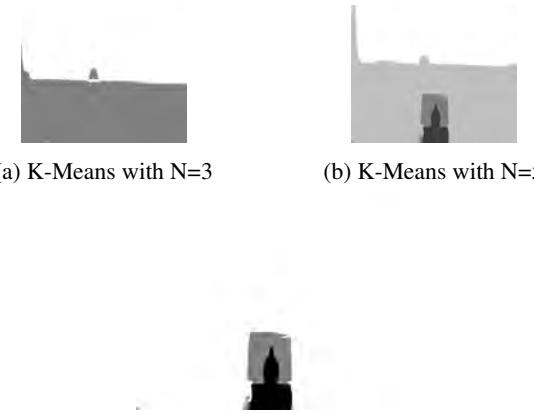
Once the holes in the original depth map have been filled, a segmentation stage is necessary to divide the depth map into a user-specified number of focus regions. These regions delineate where the focus can be in the final refocused image. However, due to the intensity distribution of pixels within the depth map, grouping the clusters of pixels into their proper distributions for *N* desired focus regions was not feasible with the clustering algorithms we tested (*e.g.*K-Means and Mean Slice Segmentation).

As a result, the user must manually over-specify how many clusters the pixel intensities of the depth map should be grouped into. The additional number of clusters is determined by the user; we found that for three desired focus regions, an additional two for the clustering step was suitable. K-Means is thus run for the over-specified number of clusters.

Once the clustering step is completed, the user must manually relabel the clusters into the desired number of focus regions. (*e.g.* For 5 clusters, two are grouped into region 1, and the remaining two are grouped separately). The newly segmented depth map now has the desire number of focus region, and must be upsampled to match the resolution of the high resolution SLR image.

6. Upsampling and Alignment of Segmented Depth Map with SLR Image

The segmented depth map must be upsampled to match the resolution of the SLR image, using Nearest-Neighbor interpolation to avoid the addition new colors into the segmented depth map. Aligned may be done algorithmically or manually in application such as Photoshop or GIMP. Given the scope of this processing pipeline and the limitations of the Microsoft Kinect and parallax between the Kinect camera and the DSLR camera, we chose to manually align them in Photoshop. The segmented depth map and the SLR image are aligned in a best fit manner; precise alignment is impossible because of the aforementioned aberrations.



(a) K-Means with N=3 (b) K-Means with N=5

(c) User-Grouped Segmented Depth Map

Figure 4: Figure 4a shows segmentation with 3 centroids, where the third clustering is almost nonexistent. Figure 4b shows that over-segmentation of the image plane allows us to define the regions of the bottle and boxes from the depth map. Figure 4c shows how user selected regions can produce a well segmented depth map

7. Depth Map Alpha Matting

At this stage of the image processing pipeline, we have an upsampled depth map aligned with its accompanying SLR image. From this stage, a simple blur kernel may be applied to the areas of defocus in the SLR image. However, the sharp edges of the segmented depth map produce unwanted halo effects. To remedy this issue, we use the concept of an alpha matte. As presented in [4], matting extracts the foreground object from an image, along with an opacity mask α from a given image $\mathbf{I}$. The image $\mathbf{I}$ is a linear combination of the foreground $\mathbf{F}$ and the background image $\mathbf{B}$ such that:

$$I = \alpha F + (1 - \alpha)B$$

The use of an alpha matte allows us to create a linear combination of foreground and background pixels. The alpha matte has values ranging [0,1] for every pixel, each pixel has a probability p_f of belonging in the foreground; 0 represents a background pixel, 1 represents a foreground pixel, and the rest are uncertain. This allows pixels on the

boundary region to receive contributions from the blur kernel and from the focused foreground region to create a more accurate shallow DoF effect. Creating an alpha matte generally requires the use of a trimap, mentioned below.

7.1. Trimap Generation

The idea behind improving the digital blur begins with smoothing out the boundaries between image planes in the depth map. With the segmented depth map, the user may pick region(s) to be in the foreground group, F, of the trimap. We use the segmented depth map as the basis for the trimap.

A trimap is a prior in the generation of an alpha matte; pixels that the user knows are certainly in the foreground are given a value of 1, while pixels that the user knows for sure are in the background, G, are given the value 0. Values that the use are unsure about (U) are given an intermediate value, .5. They can be generated automatically or hand painted. The selected region(s) for the foreground group F are not necessarily foreground by depth, but what the user decides should be the focus region.

To generate the trimap's third region, the unknown U boundary region automatically, we apply a Gaussian blur kernel to the current trimap with only two groups (since the pixels have been segmented into F and G) and then reassign any pixel that isn't 1 or 0 (definitely in F or G) with a value of 0.5 to represent the unknown boundary region. Blur kernel size may be tweaked as to not lose too much information on pixels we know to be definitely foreground or background.

7.2. Learning Based Digital Matting

We use an implementation of alpha matting called 'Learning Based Digital Matting' by Y. Zheng and C. Kambhamettu in [4] to generate the alpha matte described above. The matting process is treated as a semi-supervised local and global learning problem. A Matlab implementation provided by them was used. The alpha matte for the specified foreground focus region (chosen by the user) is thus generated according to the premise mentioned in 7.1.

(a) Near Focus Alpha Matte (b) Mid Focus Alpha Matte (c) Far Focus Alpha Matte

Figure 5: Alpha mattes for different chosen focuses. Image planes chosen to be in focus are represented by whiter pixels, while pixels chosen to be out of focus are presented by

8. Refocused Image Construction

The final image reconstruction I_F equation may be given as:

$$I_F = (I * A) + \mathcal{F}^{-1}(\mathcal{F}(I * (\mathbf{1} - A)) * \mathcal{F}(h))$$

Where $*$ represents element wise multiplication, h is the airy disk kernel, A is the alpha matte from the matting stage, I represents the original SLR image, and $\mathcal{F}(\cdot)$ represents the 2D Fourier Transform operator.

The alpha matte is used to extract the probability of a pixel in the foreground; the complement is used to extract the probability of a pixel in the background. By applying the blur kernel to the nonzero background probabilities and combining them with the foreground pixels via the alpha mattes, we are able to compute our refocused image according to the user specified focus region.

9. Results

Results of our image processing pipeline are shown in Figures 6 and 7. Results are compared to a 'ground truth' image taken with the SLR camera on a wide aperture lens to compare defocus blur between the camera's image and our digitally defocused image.

9.1. Analysis and Evaluation

Our metric for measuring the effectiveness of our proposed image processing pipeline was primarily qualitative. After seeing our results for a simple blur, we worked to make the defocusing more realistic with the second half of our image processing pipeline using trimaps and alpha mattes. Qualitatively the results for defocusing look very nice and similar to the lens blur, but holistically a lot of the false blurring of regions stems from improper segmentation of the depth map.

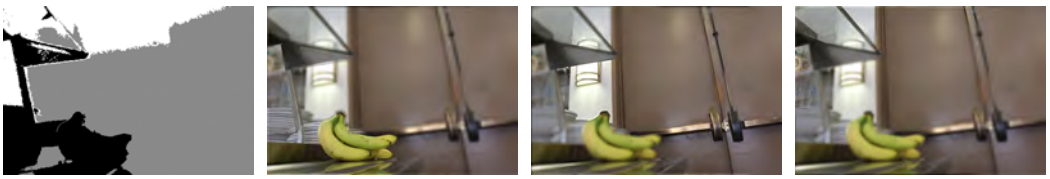
10. Conclusion

10.1. Overview

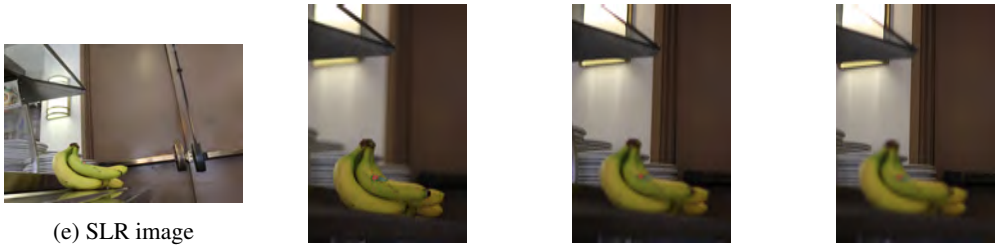
We set out to create tunable refocusing in the post-processing stage of photography. Given user-selected focus planes, we are now able to selectively refocus certain areas of the image and avoid some of the pitfalls of simpler methods. The shallow depth of field caused by the refocusing can be accentuated or diminished based on user preference. It is particularly useful for creating bold depth of field effects to assist viewers in focusing on particular focal planes in an image.

10.2. Limitations

Image Acquisition: Our method of acquiring depth information and a high resolution photograph of the

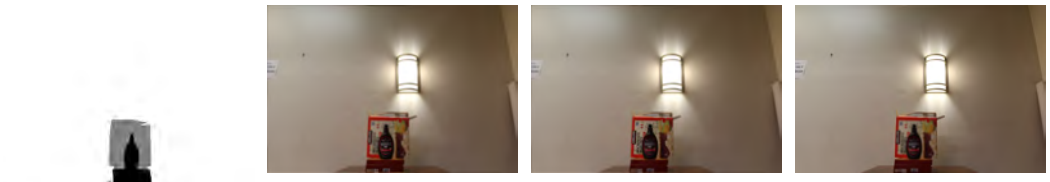


(a) Segmented depth map (b) Near focused image (c) Mid focused image (d) Far focused image

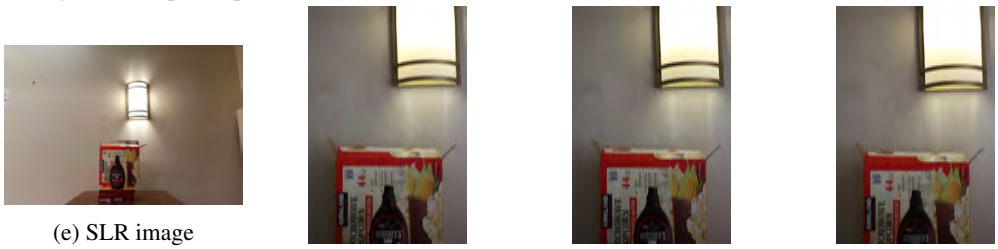


(e) SLR image (f) Near focus w/ camera (g) Mid focus w/ camera (h) Far focused w/ camera

Figure 6: Fig. 6a shows the segmented depth map after user selection of image planes. The top row shows results of defocusing on the different image planes. The lower row shows the original SLR image of the scene as well as comparison to shallow DoF by camera lens.



(a) Segmented depth map (b) Near focused image (c) Mid focused image (d) Far focused image



(e) SLR image (f) Near focus w/ camera (g) Mid focus w/ camera (h) Far focused w/ camera

Figure 7: Similar to Fig. 6, but with a different set of images.

scene required taking two images: one with the Microsoft Kinect's IR camera for a depth map, and the second with a SLR camera for a high resolution photograph. The time it takes to separately capture these photos is non trivial, and would not work with dynamic scenes. The user is limited to manually taking photos of static scenes one at a time.

Microsoft Kinect Camera: The authors used a Microsoft Kinect to acquire depth maps, via its IR camera which implements time-of-flight functionality. The Kinect V2 used for this paper has accurate depth estimation between .5 and 4.5 meters. Depths outside of that range are rendered as black (invalid) pixels. Thus, our ability to photograph certain scenes was extremely limited and had to be accomplished within that range. Occlusions were also rendered as black pixels, and had to be rectified with hole filling.

The horizontal field of view of the Kinect is 70.6° and the vertical field of view is 60°. As a result of the awkward field of view (relative to camera lenses we were able to use with the SLR) and the quality of the Kinect's IR camera, the depth map was skewed and scaled relative to the SLR photograph. This affected our ability to accurately refocus scenes and caused edge artifacts while blurring specific planes. IR Cameras do not function well with transparent objects, and they should be avoided while using this

method. IR light is not properly reflected by transparent objects, so the depth of the transparent object in the depth map is misrepresented by whatever managed to properly reflect the pulse.

There is no linear 1:1 mapping between pixel intensity and depth in the depth maps acquired via a Microsoft Kinect v2. Research was to map the pixel intensities to actual depth on the original Microsoft Kinect, but due to differences in the IR Camera Array, such research was unusable.

Alignment: Our method requires manual alignment and upsampling of the depth map gathered after the depth segmentation and region selection step specified above. Once the depth map has been upsampled, both the high resolution SLR image and the upsampled depth map must be imported back into the workspace. The authors believed that automated alignment, given the aforementioned issues with the Kinect, would be out of the scope of this project.

Scene Lighting: Taking photographs of a scene in most lighting situations with a SLR camera is easily done. However, due to limitations on the Microsoft Kinect's IR camera mentioned above, this method implemented with a Microsoft Kinect only works in scenes with minimal sunlight in order to avoid IR interference. Time-of-flight cameras that don't use IR, or are able to mitigate IR interference from the sun, would be able to compensate for interference from sunlight.

Depth Independent Blurring: Since the blur kernel is applied entirely to the regions not designated in focus, the blur is not applied on a per-depth basis, which is not entirely realistic for a shallow DoF effect. The method can be modified to accommodate for this if necessary via selective alpha matting.

Image Clustering and Segmentation: K-Means and other segmentation algorithms for 2D image segmentation (*e.g.* Mean Slice segmentation) we tested for this paper group clusters based on grayscale pixel intensity in the depth map. [1] Even though it was clear that certain depths of pixels should be grouped together, clear mis-clustering of intensities for a desired number of focus regions ($\mathbf{N}$) force the user to intervene. As a result, the user must over-segment by $\mathbf{Q}$ clusters, for a total of $\mathbf{N} + \mathbf{Q}$ clusters as the input to K-means. After, the user must re-label the clustered depth map back into the desired $\mathbf{N}$ clusters. For the method we've developed, this should be done manually.

10.3. Future Work

Testing this method with better range data is imperative. It will produce better results and less artifacts during the alpha matte generation process, and the blurring will suffer less at the edges. Additionally, the blur is unlikely to apply to regions outside of the desired focus region, creating a more realistic and less error-prone approach. Furthermore, using more test images will allow us to develop insights on tuning our kernals and K-Means segmentation, and perhaps allow us to train a new learning algorithm to dynamically find the best segments.

Ultimately, we would like to see a similar technique implemented on a rig or camera capable of taking a high resolution picture of the scene and acquiring the depth information from the scene simultaneously. The depth information can be acquired via structured illumination and a powerful flash as suggested in [3], or through a powerful IR camera that can minimize the effects of sunlight. By having both the high resolution shot-taking and the depth mapping occur in parallel, minimal alignment would be required given a setup that accounts for distortion between the two image acquisition methods. Given the recent emphasis on hobbyist photography by modern phone designers (e.g. the Apple iPhone 7+ two cameras for depth-of-field effects), the authors would like to see post-processing refocusing occur on smartphones in the near future.

[1] Due to aberrations in the depth data caused by the Microsoft Kinect v2, we felt it would be infeasible to combine the color data with the depth data for more accurate segmentation.

11. Acknowledgements

We thank Professor Gordon Wetzstein for a great class and Timon Ruban for help with the depth map acquisition. We also thank Yuanjie Zhang and Chandra Kambhamettu for providing their implementation of Learning Based Digital Matting in [4] which was critical for achieving more accurate image reconstruction, Anton Semechko for a fast and efficient Matlab implementation of K-Means for grayscale images, and Daniel Shiffman for his open source implementation of OpenKinect for Processing.

References

[1] B. Huhle, T. Schairer, P. Jenke, and W. Straßer. Realistic depth blur for images with range data. In *Dynamic 3D Imaging*, pages 84–95. Springer, 2009.

[2] S. Liu, C. Chen, and N. Kehtarnavaz. A computationally efficient denoising and hole-filling method for depth image enhancement. In *SPIE Photonics Europe*, pages 98970V–98970V. International Society for Optics and Photonics, 2016.

[3] F. Moreno-Noguer, P. N. Belhumeur, and S. K. Nayar. Active refocusing of images and videos. *ACM Transactions On Graphics (TOG)*, 26(3):67, 2007.

[4] Y. Zheng and C. Kambhamettu. Learning based digital matting. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 889–896. IEEE, 2009.