Analysis of preprocessing and rendering light field videos

Christian Petry

Since light field cameras became available on the consumer market, its versatile functions are more and more known to computer vision communities and photographers. After all, refocusing an image after it was taken is one of the features most wished for in photography. The possibility of refocusing a video after being taken extends this feature to an additional dimension, enabling direct user interaction in videos, which could result in a new form of entertainment and user experience.

This work presents different approaches of preprocessing and rendering light field videos made with a Lytro Illum, a plenoptic camera using microlenses. We compare three differently preprocessed light field videos and reveal the main issues for rendering on consumer hardware. Our results show, that one of the biggest obstacles lies in the transfer of data from light field video files to the main memory of the system. Therefore, the demosaiced and color corrected representation that balances preprocessing and resulting video file size is fastest in rendering light field video files. Rendering a short light field video directly from the main memory allows interaction like refocusing in real time and points out the need for future research on new codecs for light field videos based on non-raw light field images.

Additional Key Words: Light fields, videos, preprocessing, rendering performance

1 INTRODUCTION

An image taken by a conventional camera captures light heading towards a camera on a 2D plane. A light field camera is able to additionally collect spatial and angular light information. This enables applications like refocusing and view perspective shifting [Adelson and Wang (1992), Ng (2006)]. With a robust eye tracking, it could be even possible to adjust focus by simply looking at different areas in the scene. The possibility of view perspective shifting also enables surface reconstruction [Tao et al. (2013)].

The company Raytrix was the first to build and market commercially available plenoptic cameras [Zhang (2010)]. Their main use cases focus on microscopy, the manufacturing and automotive industry. Lytro Inc., founded by Ren Ng in 2006, is a company selling light field cameras on the consumer market. Lytro Inc. has brought out two versions of hand held cameras, which can capture light fields and display them on a desktop pc, as well as on the web [Lytro (2015)]. With the Lytro desktop application the user can not only change conventional settings like color correction, sharpening and denoising, but is also able to refocus, extract depth maps or slightly shift view perspective.

C. Petry
University of Applied Sciences Würzburg-Schweinfurt,
Würzburg, Deutschland,
http://www.fhws.de

Sequential light fields, or light field videos, aren't yet manageable by Lytro's cameras and software. The German company Raytrix however, built and marketed light field video cameras specially made for industrial use cases. Their prices though are unaffordable for end-users in the consumer market [Raytrix (2013)]. Even these cameras can only handle 7 frames per second at a resolution of only 10 megapixel [Raytrix (2015)]. In the year 2012 Raytrix presented a light field video camera at the "GPU Technology Conference" (GTC) [Perwass (2012)]. However, on a scientific point of view, the topic of light field videos with a single microlens camera is still quite unexplored.

In this work we analyze different approaches of preprocessing and rendering light field videos made with the Lytro Illum, a consumer light field video camera. As the Lytro Illum can only take one picture at a time, we analyze the possibilities on stop motion videos. We execute three sequential calculation steps when decoding a Lytro light field image. The first step is the extraction of raw light field images from files created by the camera. Second is demosaicing as well as color correcting the raw image and third is the rearrangement of the microlens image as a stitched image representation, similar to that of a 2D image array. With our framework, these three steps can either be executed beforehand or calculated on demand when rendering the video. After preprocessing it is possible to either save the light fields as consecutive images or as a compressed video on hard drive. In our evaluation, we analyze possibilities, restrictions and performance issues of rendering interactive light field videos based on these different preprocessing steps.

The paper is organized as follows: Section 2 reviews relevant work and the light field concept. In Section 3 we describe the three consecutive preprocessing steps, which we then use in Section 4 to oppose different approaches on rendering interactive light field videos. Finally, Section 5 then adds a conclusion of our results and further thoughts on light field videos.

2 RELATED WORK

The light field, as it was first introduced by A. Gershun in his paper, is defined by a plenoptic function in a static context [Gershun et al. (1939), Adelson and Bergen (1991)]. In light field videos though, the 4D function of a light field reclaims time as an additional parameter from the original plenoptic function. Since the year 1996, a free archive of light fields has been created at Stanford with a Multi-Camera array, a Lego Mindstorms gantry and a light field microscope, and was updated ever since [Levoy and Hanrahan (1996b)].
Several light field video systems have been developed based on camera arrays arranged on a rectangular plane [Chan et al. (2005); Wilburn et al. (2005)]. A hemispherical arranged camera system has been presented by [Akin et al. (2013)]. B. Smith described a stabilization technique for shaky light field videos [Smith et al. (2009)]. All these systems though, use several conventional cameras to capture the scene from different directions and do not include the possibilities of microlenses.

Recently, a light field video captured by a microlens camera, led to a 5-D depth velocity digital filter to enhance moving objects in light field videos [Edussooriya et al. (2015)].

3 DECODING AND DISPLAYING A LYTRO LIGHT FIELD

The Lytro Illum from Lytro Inc. is a camera, that uses microlenses to capture a light field (see figure 1). In total the CMOS sensor is able to capture about 40 megapixels (also called "Megarays") with a main lens that enables 8x optical zoom and 30-250mm equivalent focal length [Lytro (2015)]. In between the main camera lens and the photo sensor is an array of very small lenses. Each lens captures a tiny fraction of the scene viewed by the camera. Due to the main lens in front of the microlens array, the scene is viewed with the same field of view by each microlens. Each raw light field image taken with the Lytro Illum has a resolution of $7728 \cdot 5368$ pixels. With each pixel encoded in 10 bits and several additional meta information, the light fieldfile created by the camera needs approximately 53 MB disk space.

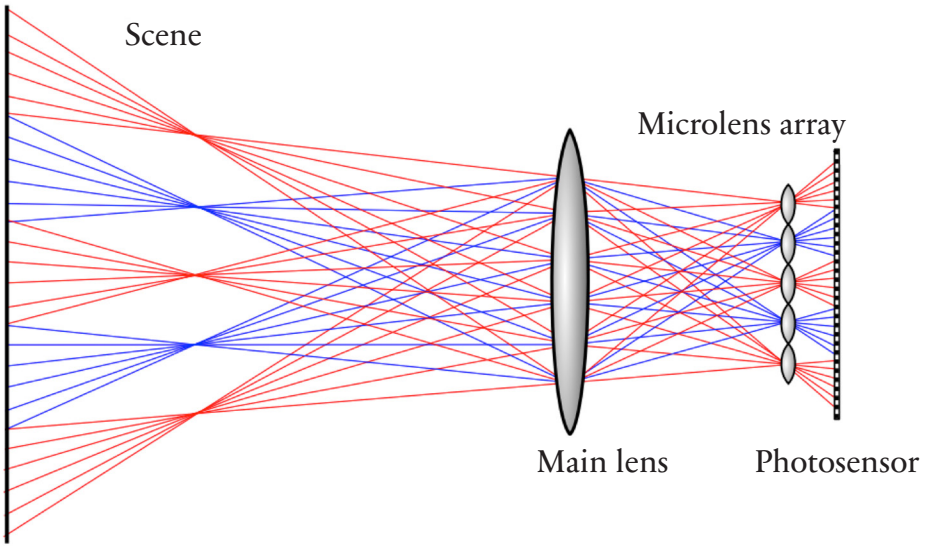


Fig. 1: Capturing a light field with a microlens array and a main lens. Each microlens records a small view of the scene (Source: Own Source)

In this section, three steps of preprocessing a light field are described. These steps are calculated sequentially and can be distributed as follows:

(1) Extracting the raw light field from the container file.
(2) Demosaicing and color correction.
(3) Decoding the microlens array to a convenient representation for rendering.

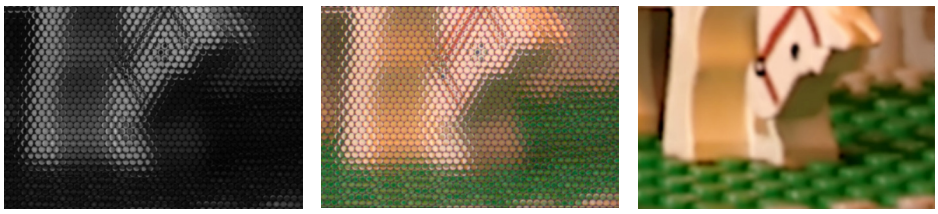
Each of these steps are coded in our software framework and their results can either be preprocessed and saved as image sequences (videos) to a hard drive, or calculated on demand by the GPU inside our glsl shader, when rendering the video frame by frame. The rendered result is always the same, independent of the amount of preprocessing steps. In the end, the result can be displayed as a video, which focus can be interactively manipulated by the viewer. Our software is available in our repository at https://bitbucket.org/cpetry/lfp-reader.

3.1 Extracting the raw light field

After a light field was taken, the Lytro Illum creates a file of type *.lfp, containing different several kinds of information, such as the light field, camera parameters and calibration details. Lytro support employees themselves are directing towards the Light Field Toolbox by Donald

Danserau [Dansereau (2012)] because "...at this point, there doesn't exist a published documentation on the file format". Other approaches to decode the container are outdated or not open source and do not work with the newer version of the Lytro Illum. Still, they are good enough to get a small impression on how Lytro saves light field images [Kucera (2015); Patel (2012)]. For our evaluation though, we created a new framework for decoding light fields.

As mentioned in the meta information inside the *.lfp file format, the raw light field is encoded in a 10 bit per pixel single channel image. Saving this raw image in a PNG lossless format, reduced from 10 to 8 bit per pixel, results in the first step of our decoding process (see figure 2(a)). Reducing the bits per pixel is necessary for displaying the whole color range on a standard computer monitor. As seen in the cropped and zoomed figures 2(a) and 2(b), the light field image is made of a great number of microlenses. In the raw representation, it is still possible to adapt color correction, gamma values and other post processing steps done in conventional photography when rendering from this stage on. It is essential to keep this in a lossless image format because of consequential errors regarding further decoding steps. This results in a grayscale image with 8 bits per pixel.



(a) Extracted raw light field

(b) Demosaiced and color corrected

(c) A completely decoded light field

Fig. 2. Zoomed representation of light field images, showing the first two steps of decoding a light field (Source: Own Source)

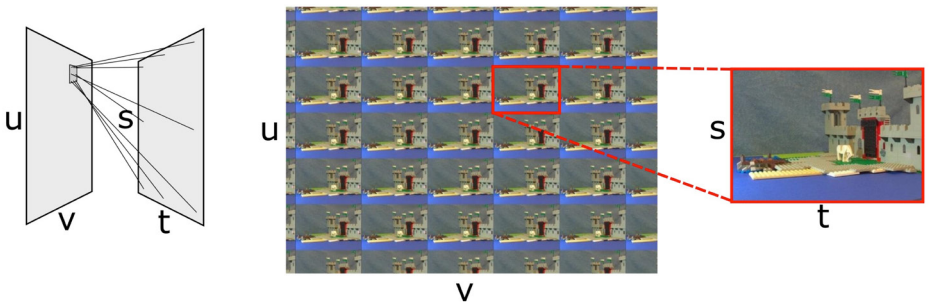


Fig. 3: UVST representation of a light field. One coordinate on the uv plane represents one image on the st plane. Here the uv plane is cropped. This representation is derived from [Levoy and Hanrahan (1996a)]

3.2 Demosaicing and color correction

In the second stage, the raw image has to be demosaiced. Inside the meta information are details about the Bayer color filtering of the raw format [Bayer (1976)]. The current Lytro Illum files are filtered with a "grbg"-filter. That means, that the colors are aligned in blocks of 4 pixels. The upper left and lower right pixel is green, the upper right is blue and the lower left represents a red color value. From these values it is possible to reconstruct coloring while maintaining image resolution, which is called demosaicing. Lytro does not publish any infor-

mation on their demosaicing algorithm of the Lytro desktop application. Therefore, we chose for our analysis a bilinear bayer interpolation. Other interpolation methods could enhance visual quality, but comes with a much higher computational cost [Malvar et al. (2004)].

After demosaicing the image has to be color corrected. Information about white balance, color correction matrix and gamma values are extracted from the meta file sections of the light field file. The demosaiced and color corrected image, as seen in figure 2(b) defines the second step of our decoding process. This file now consists of 24 bits per pixel with 3 color channels.

3.3 Decoding the microlens array for display

The next step is a conversion from microlenses to a 2d image array. The main purpose for this representation is the fast linear interpolation done by the graphics card [Woo et al. (1999)]. This enhances visual quality by smoothing pixel borders in small resolution images. For constructing the convenient representation $\Psi(u, v, s, t)$, (see the cropped image in center of figure 3) [Levoy and Hanrahan (1996a)], several additional information have to be recovered. The microlens center offset position in pixels, the clockwise rotation of the microlenses in degrees, and the lens and the pixel pitch of the microlens-array in meters. The algorithm for decoding the 2D sensor image to a 4D light field is further described by [Dansereau et al. (2013)]. By using this algorithm, the resolution of the light field image increases, because of reusing pixels at the border of each microlens. The resulting image is hereafter called UVST representation, which has $15 \cdot 5$ sub-images of $625 \cdot 433$ pixels each. Every sub-image shows a different view of the same scene. As a summary, the result of each preprocessing step is shown in the table below:

Preprocessing stage	Image size	Bpp	Size per light field
Raw grayscale	$7728 \cdot 5368$	8 bit	40.30 MB
Demosaiced & color corrected	$7728 \cdot 5368$	24 bit	120.90 MB
UVST representation	$9375 \cdot 6495$	24 bit	174.21 MB

Table 1: Summary of preprocessing steps and their characteristics

In our approach, we used the UVST representation as the basis for rendering light fields. All the images on the st plane are displayed as off-axis (sheared) perspective views of the scene (see center image of figure 3) [Levoy and Hanrahan (1996a)]. One pixel (x, y) at a specific coordinate (u, v) is then defined by

$$(x, y)=(s, t)+(u, v)\cdot s_{st}$$

with s_{st} as the size of the st plane, which equals to that of a sub-image with $625 \cdot 433$ pixels each.

3.4 Rendering the light field

Displaying a focused image of the light field, is then achieved by interpolating different shear warped (u, v) sub-images. Each neighboring sub-image inside a given radius of a given (u, v) coordinate gets slightly translated and added to the final image. Depending on the translation different parts of the scene are focused. Higher translation in direction of the center sub-image results in an image focused on objects further away from the initial focal length. A translation away from the center, produces an image focusing objects closer to the camera. The higher

[1] Bpp: bits per pixel.
[2] Sizes are stated as uncompressed images.
[3] The resolution grows due to the algorithm that reuses a one pixel border of each lens [Dansereau et al. 2013].

the chosen radius of sub-images used for this, the blurrier the image gets in parts not focused on. This idea was taken from the aperture viewer of the Stanford light field archive [Vaish and Adams (2008)].

After all preprocessing steps, a focused image of a Lytro Illum in its original state has a resolution of $625 \cdot 433$. This at first seems to be a startlingly low resolution, but can be increased by using super resolution algorithms [Bishop and Favaro (2012)]. This technique can render an image to three or four times the original size. For this work though, we evaluate videos in their original resolution, to not add optional factors influencing performance.

An overview of the here described steps, their amount of preprocessing and left over rendering workload, is visually compared in figure 4.

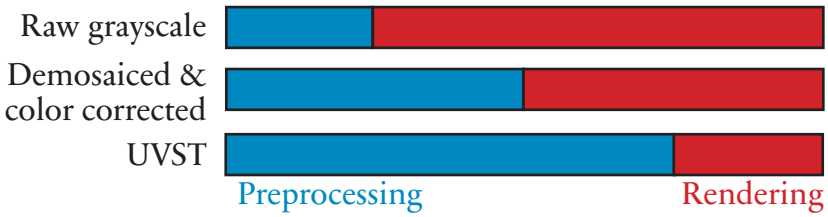


Fig. 4: Amount of preprocessed (blue) and left over rendering workload (red) for each representation described above. This visualization does not represent correct workload dimensions and only shows an overview for better understanding of further analysis (Source: Own Source)

4 FRAMEWORK ANALYSIS AND EXPERIMENTAL RESULTS

In order to render a video on consumer hardware, the frames have to be copied consecutively into the graphic cards memory. One straight forward solution is a pipeline that starts with decoding a frame from a video file stored on a hard drive and copying the result to the main memory. Afterwards the image gets forwarded to the graphic cards memory, where it is then further processed for rendering the interactive video.

For our test purposes, we shot a light field sequence of 116 frames with a Lytro Illum, resulting in a short stop motion video. The duration and thus the frame rate of the video, is determined by the processing speed of the whole system and the preprocessing stage the video is in. In this section, we determine bottlenecks of the processing pipeline to derive future possible enhancements. First we compare the preprocessing steps described above when rendering a video file directly from the hard drive. In the second step we analyze playing back a video completely copied onto the main memory. To get direct rendering workload of the graphics card, the third part analyzes pure rendering performance directly from the memory of the graphics card without any data transfer.

4.1 Comparison on rendering a video file

Playing back video files with a resolution of $1920 \cdot 1080$ (Full HD, 1080p) and even $4096 \cdot 2160$ (4K) does not pose a problem for current hardware. Additionally encoding a video file with H.264 (also called MPEG-4 AVC) or H.265 (also called HEVC) reduces the size of the resulting video file but comes with additional decoding calculations [ITU-T (2014a); ITU-T (2014b)]. These codecs though, do not work with a full resolution Lytro Illum light field. As stated in the re-

commendations, the maximum resolution for H.264 is 4096 · 2304 and for H.265 is 8192 · 4320 pixels. A lossy compression would furthermore change neighboring pixel values, which would result in incorrect colors, when demosaicing frames from the raw grayscale representation. To reduce the size of light field videos anyway, we use HuffYUV, one of the fastest and best performing lossless codecs for decoding and encoding videos [Lab (2007); Hashemian (1995)].

Preprocessing stage	Video file size	FPS	Processed MB/s
Raw grayscale	4.48 GB	4.0 – 4.3	~170 MB/s
Demosaiced & color corrected	6.11 GB	4.3 – 4.6	~242 MB/s
UVST representation	7.41 GB	3.5 – 3.8	~248 MB/s

Table 2: Performance comparison rendering HuffYUV encoded video files of a 116 frames stop motion video

The compressed size, rendering frame rate and processed MB/s are displayed in table 2. Notice that the raw video file does not have the approximate 1/3 file size of the demosaiced file. This is due to the HuffYUV compression algorithm. The video file was played back by a Intel Core i5-4460 CPU @ 3.20Ghz with an ATI Radeon 5850. The file was read from an up to date Solid State Disk resulting in 248 MB/s as maximum processed data achieved. To copy frames as fast as possible asynchronous Pixel Buffer Objects were used.

The fastest representation is the demosaiced & color corrected version of the video. One cause of this result is the lower frame size in comparison to the UVST representation. Approximately only 68 % of pixels of the UVST representation have to be copied. The raw grayscale approach in comparison has much more decoding and texture lookups left to do but still outperforms the UVST representation, due to much less data needed copying.

4.2 Comparison on rendering a preloaded video on the main memory

One of the main bottlenecks of rendering the video, derived from the results of table 2, is the transfer of data from the hard drive to the main memory. To be independent of the hard drive reading speed, the light field video can be rendered directly from the main memory. By reading the complete image sequence into the main memory before starting to render the video, both, video decoding and hard drive speed restrictions, can be avoided. However, this approach is only possible for short sequences or systems with a very high amount of main memory. The stop motion video (116 frames) we used, needs a system with more than 16 GB memory to be able to render it completely, depending on the representation.

Preprocessing stage	RAM usage	FPS	Processed MB/s
Raw grayscale	17.39 GB	8 – 9	~1424.25 MB/s
Demosaiced & color corrected	17.39 GB	15 – 16	~2531.98 MB/s
UVST representation	26.31 GB	20 – 21	~4877.87 MB/s

Table 3: Performance comparison of rendering sequential images residing on the main memory.

Frame rates and actual memory usage are shown in table 3. Now that the restriction of hard drive reading speed is avoided, the UVST representation can be rendered nearly fluently with about 20 FPS. It outperforms in terms of processed MB/s every other representation, but needs the most space on the main memory. The memory needed for all frames is by far higher than in the previous performance comparison. Each image needs now four color channels (RGBA)

and is uncompressed. Saving every frame on RAM as a four color channel image, increases the rendering performance by more than 300% in comparison to saving it as a three color channel image (Demosaiced & color corrected with only RGB: 4-5 frames). This significant difference is the cause of using optimal pixel formats of graphics cards [Nvidia (2007)].

4.3 Comparison on rendering one frame consecutively

When rendering the light field, the amount of preprocessing has a big impact on the frames per second shown on the screen. The more work has been done in advance, the less calculations the GPU has to do for each rendering cycle. Here, we compare rendering only one frame over and over, to get the maximum average rendering frame rate without any data transfer whatsoever. The results are shown in table 4.

Preprocessing stage	Image size	FPS
Raw grayscale	7728 · 5368	8 – 9
Demosaiced & color corrected	7728 · 5368	25 – 26
UVST representation	9375 · 6495	60 – 63

Table 4: Frames per second repeatedly rendering one single light field image

This shows the performance of changing focus of an image on a single light field. Computing a focused image from the raw representation needs too many calculations to be rendered fluently on current hardware. The frame rate remains between 8 and 9 frames per second. The demosaiced & color corrected representation can be rendered in realtime (above 24 FPS). Although the light field image is the largest in the UVST representation, the frame rate achieved clearly dominates. With rendering at 60 FPS a very fast and fluent user interaction can be achieved, which behaves similar to refocusing a scene with a camera before taking a photo. At this frame rate it is possible to reconsider super resolution or other additional visual improvements.

5 CONCLUSION

We presented a framework, which can render interactive light field videos on basis of different preprocessing steps on current standard consumer hardware. In every representation the video can be refocused directly by user input. The representation with the least preprocessing shown here, addressed as raw grayscale, enables additional interactions, such as color correction, adjusting gamma values or blurring and sharpening. These interactions can be compared to post processing of raw images taken with a conventional camera.

The frame rate though, when playing back a light field video file, is not high enough on current consumer hardware to realize an application for real time usage. Neither of the here presented preprocessing steps achieved more than 5 frames per second while rendering. The best performance achieved is that of a demosaiced & color corrected representation with about 4.5 frames per second. Derived from further analysis, one of the main tasks is a fast transfer of frames from a file on the hard drive to the memory of the graphics card.

When rendering directly from main memory, less decoding steps significantly improve rendering performance. The representation just before the final rendering, here called UVST representation, is the fastest and also the biggest in terms of file size, but can be used for near real time (over 20 FPS) interactive rendering when stored completely on the main memory. This requires either huge amount of memory or more advanced and adapted encoding and decoding algorithms.

As light field cameras are still quite new, resolution and the size of microlenses can still vary with newer technologies. A possible coding algorithm, therefore should be independent of several of these parameters. In consideration of previously developed codecs, one solution could be a new encoding algorithm applied to an UVST representation. This would maximize rendering performance and minimize artifacts as well as the file size of the video. Encoding a demosaiced & color corrected representation with an adapted rendering algorithm could pose a worthy alternative, which by nature has less pixel than the UVST representation and thus uses less bits per image.

REFERENCES

Adelson, E. H. and Bergen, J. R. (1991): The plenoptic function and the elements of early vision, in: Landy, M. S. and Movshon, J. A. (Eds.) Computational models of visual processing, Cambridge, MA: MIT Press, 1991, P. 3-20.

Adelson, E. H. and Wang, J. Y. A. (1992): Single lens stereo with a plenoptic camera, in: IEEE transactions on pattern analysis and machine intelligence 14, 2, 1992, P. 99-106.

Akin, A., Cogal, O., Seyid, K., Afshari, H., Schmid, A., and Leblebici, Y. (2013): Hemispherical multiple camera system for high resolution omni-directional light field imaging, in: Emerging and Selected Topics in Circuits and Systems, IEEE Journal on 3, 2, 2013, P. 137-144.

Bayer, B. (1976): US Patent 3,971,065, Color imaging array, http://www.google.de/patents/US3971065, accessed July 6th, 2015.

Bishop, T. E. and Favaro, P. (2012): The light field camera: Extended depth of field, aliasing, and superresolution, in: Pattern Analysis and Machine Intelligence, IEEE Transactions on 34, 5, 2012, P. 972-986.

Chan, S.-C., Ng, K.-T., Gan, Z.-F., Chan, K.-L., and Shum, H.-Y. (2005): The plenoptic video, in: Circuits and Systems for Video Technology, IEEE Transactions on 15, 12, 2005, P. 1650-1659.

Dansereau, D. (2012): Light field toolbox v0.4, http://www.mathworks.com/matlabcentral/fileexchange/49683-light-field-toolbox-v0-4, accessed July 6th, 2015.

Dansereau, D., Pizarro, O., and Williams, S. (2013): Decoding, calibration and rectification for lenselet-based plenoptic cameras, in: Computer Vision and Pattern Recognition (CVPR), 2013, P. 1027-1034.

Edussooriya, C., Dansereau, D., Bruton, L., and Agathoklis, P. (2015): Five-dimensional depth-velocity filtering for enhancing moving objects in light field videos, in: Signal Processing, IEEE Transactions on 63, 8, 2015, P. 2151-2163.

Gershun, A., Moon, P. H., and Timoshenko, G. (1939): The light field, Massachusetts Institute of Technology, 1939.

Hashemian, R. (1995): Memory efficient and high-speed search huffman coding, in: IEEE Transactions on Communications, 43, 10, 1995, P. 2576-2581.

ITU-T. H.264 (2014a): Advanced video coding for generic audiovisual services, https://www.itu.int/rec/T-REC-H.264-201402-I/en, accessed July 6th, 2015.

ITU-T. H.265 (2014b): High efficiency video coding, https://www.itu.int/rec/T-REC-H.265-201504-P/en, accessed July 6th, 2015.

Kucera, J. (2015): Lytro meltdown, http://optics.miloush.net/lytro/, accessed July 6th, 2015.

Lab, C. M. G. (2007): Lossless video codecs comparison '2007, http://compression.ru/video/codec_comparison/pdf/msu_lossless_codecs_comparison_2007_eng.pdf,accessed July 6th, 2015.

Levoy, M. and Hanrahan, P. (1996a): Light field rendering, in: SIGGRAPH '96, Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, New York, 1996, P. 31-42.

Levoy, M. and Hanrahan, P. (1996b): The (old) stanford light field archive, http://graphics.stanford.edu/software/lightpack/lifs.html, accessed July 6th, 2015.

Lytro, I. (2015): Lytro Illum, https://www.lytro.com/illum/, accessed July 6th, 2015.

Malvar, H. S., He, L.-W., and Cutler, R. (2004): High-quality linear interpolation for demosaicing of bayerpatterned color images, in: International Conference of Acoustic, Speech and Signal Processing, Institute of Electrical and Electronics Engineers, Inc, 2004, P. 5-8.

Ng, R. (2006): Digital light field photography. Ph.D. thesis, stanford university.

Nvidia (2007): Fast texture downloads and readbacks using pixel buffer objects in OpenGL, http://http.download.nvidia.com/developer/Papers/2005/Fast_Texture_Transfers/Fast_Texture_Transfers.pdf, accessed July 6th, 2015.

Patel, N. (2012): lfptools, https://github.com/nrpatel/lfptools, accessed July 6th, 2015.

Perwass, C. (2012): Live 3d-video with a lightfield camera, http://on-demand.gputechconf.com/gtc/2012/presentations/S0335-Live-3D-Video-with-a-Lightfield-Camera.pdf, accessed July 6th, 2015.

Raytrix. (2013): 3d light field camera technology, b2b reseller price list, http://www.raytrix.de/tl_files/downloads/products.pdf, accessed July 17th, 2015.

Raytrix. (2015): R42 SERIES, http://www.raytrix.de/produkte/#r42series, accessed July 17th, 2015.

Smith, B., Zhang, L., Jin, H., and Agarwala, A. (2009): Light field video stabilization, in: IEEE 12th International Conference on Computer Vision (ICCV), 2009, P. 341-348.

Tao, M. W., Hadap, S., Malik, J., and Ramamoorthi, R. (2013): Depth from combining defocus and correspondence using light-field cameras, in: IEEE International Conference on Computer Vision (ICCV), 2013, P. 673-680.

Vaish, V. and Adams, A. (2008): The (new) stanford light field archive, http://lightfield.stanford.edu/lfs.html, accessed July 6th, 2015.

Wilburn, B., Joshi, N., Vaish, V., Talvala, E.-V., Antunez, E., Barth, A., Adams, A., Horowitz, M., and Levoy, M. (2005): High performance imaging using large camera arrays, in: ACM Trans. Graph. 24, 3, 2005, P. 765-776.

Woo, M., Neider, J., Davis, T., Shreiner, D., et al. (1999): Open GL programming guide, Boston, 1999.

Zhang, M. P. (2010): The First Plenoptic Camera on the Market, http://petapixel.com/2010/09/23/the-first-plenoptic-camera-on-the-market/, accessed July 6th, 2015.