Development of Simulation Tools for Exploring Plenoptic Camera Applications

CJ Meah[1,2] , IB Styles[2]

[1]PSIBS DTC, University of Birmingham, UK.

cjm861@bham.ac.uk

[2]School of Computer Science, University of Birmingham, UK.

Abstract

Plenoptic camera design has many parameters to optimise in order to achieve the best possible 3-D imaging capability. Furthermore construction of the cameras requires specific elements to be placed extremely accurately in the optical train of the system, and optimal parameters differ between applications. An important tool for design and development is simulation, which allows exploration and optimisation without physically committing to a system. Rendering and modelling techniques for plenoptic data are also important in extracting the maximum amount of information, and can be performed on the simulated data. Simulation and rendering software is being developed to allow for *in silico* exploration.

1 Introduction

An ordinary camera consists of a lens and a sensor, and generates an in focus image for the imaging plane of the system, with all other planes out of focus in the final image. In order to change viewpoint, aperture, or focal point, the system must be physically changed and another image acquired. The basis of a plenoptic camera is to preserve the directional information of rays which is lost in a conventional camera; rays which would have met on a sensor pixel are split onto separate pixels by a microlens.

By sacrificing spatial resolution for this angular information image viewpoint, focal plane, and aperture can be changed computationally post-acquisition. In essence, plenoptic cameras provide a 3-D representation of a scene from a single acquisition. This ability has a wealth of applications in a variety of areas.

One issue that a number of free parameters dictate the final performance of the plenoptic system, meaning optimisation of these parameters is needed. Microlens properties also play an important role in determining system performance, and microlens array placement is key to a trade-off in spatio-angular resolution, and often must be accurate on the order of microns. Furthermore, the components of the system must complement each other to attain the maximum efficiency in imaging, for example matching the f-number of both the microlenses and the main lens. These parameters are difficult to explore physically, which is where simulation tools can aid design and exploration.

2 Plenoptic Architectures

There are two well established plenoptic architectures which take different approaches to tackling the spatio-angular trade-off (Georgiev et al. 2006). The original plenoptic camera (often denoted as Plenoptic 1.0) (Ng et al. 2005) consists of a microlens array placed at the principle image plane of the main lens, and the image sensor one focal length behind the microlenses. The microlenses are completely defocused from the scene, and each microlens gathers information for a single conjugate spatial point. In a final rendered image each microlens therefore contributes to a single pixel, meaning that spatial resolution is determined by the number of microlenses in the array. Computational rendering strategies can improve on this single pixel contribution per microlens (Lumsdaine & Georgiev 2008), but it is a useful baseline. The rays hitting each microlens are split onto the sensor pixels, which can be seen as angular binning. Therefore, the number of pixels covered by a microlens determines the angular resolution.

The emphasis of the original plenoptic camera is on gaining angular resolution, however this is at the expense of spatial resolution which is often the priority in many applications. An alternative plenoptic set-up called the focused plenoptic camera (or Plenoptic 2.0) (Todor Georgiev & Lumsdaine 2010) trades back this gain in angular information for spatial information. It uses the microlenses as a relay system, placing the microlens array a distance ***a*** from the image plane and the sensor a distance ***b*** from the microlens plane, satisfying

$$1/f = 1/a + 1/b ,$$

where f is the microlens focal length. The angular distribution for a spatial point is now spread over multiple microlenses, and each micro-image represents a spatial image. The raw image produced by the system can be pictured as taken by a multi-view stereo system. When rendering a final image, patches can be taken from micro-images and tiled together, meaning many more pixels per microlens contributing to the image compared with the original plenoptic camera.

Table 1: Outline of different plenoptic architectures.

	Plenoptic Camera Properties	
Properties	Original Plenoptic Camera	Focused Plenoptic Camera
Microlens Plane (**mP**)	Main Lens Image Plane (**MLIP**)	**MLIP** + ***a***
Sensor Plane	**mP** + microlens focal length	**mP** + ***b***
Spatial Resolution	# microlenses	***b/a*** of the sensor resolution
Angular Resolution	# pixels under microlens	***a/b***

Both plenoptic architectures perform differently due to their approach to the spatio-angular trade-off. It is useful to consider both of these choices; however the physical set-up and component specification will be different for both, which is why plenoptic simulations are useful.

3 Plenoptic Simulation

Optical simulation software is widely accessible, with many commercial packages available. However these are often very expensive and are not specifically designed for plenoptic image exploration. Therefore a software tool was created to enable any architecture of plenoptic camera to be simulated with an input scene.

The majority of the simulation utilises ray tracing methods for propagating geometric rays through the optical system, which consists of an arbitrary set of definable optical components. Ray transfer matrices were used for this purpose since they are easy to define, extend, and can be calculated very quickly. In a ray transfer matrix system, a ray is defined by a height $\boldsymbol{h}$ and angle $\boldsymbol{\theta}$, which are both relative to the optical axis. By defining a ray in these terms, it can be operated on by a matrix describing an optical path. For instance, the free space and lens matrices are defined as:

$$T = \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & 0 \\ -1/f & 1 \end{bmatrix},$$

where $\boldsymbol{d}$ is the distance travelled in free space, and $\boldsymbol{f}$ is the focal length of the lens. These matrices can be combined so as to produce a system matrix, meaning that a single operation now propagates the ray through the system. In terms of simulating a microlens array, each microlens defines its own optical axis, and it is not known before tracing which lens a ray will hit. Therefore a separate step is involved, where a ray is traced to its intersection with the microlens array and then operated on by the particular microlens it hits. The microlens centre must be treated as the optical axis for the lens refraction operation, but then returned to the system axis.

Instead of considering each ray individually, we can form a vector of rays and operate on this whole vector at once. This is useful in speeding up computation on a CPU. Since plenoptic cameras sacrifice spatial resolution for angular, a higher resolution sensor is normally needed to alleviate these effects. However, a higher resolution sensor means many more rays than are usually needed. This, along with the need to sample angular information adequately, means that a plenoptic simulation needs many more rays than are usually considered in standard ray tracing applications. For a realistic simulation often vectors containing millions of rays are needed.

A key issue in the simulation software is computational speed and efficiency. In backwards ray tracing, a ray is propagated from the sensor through the optical system to the object. This has the advantage of only considering rays which are guaranteed to hit the sensor, which means much higher efficiency for dense scenes and in general. Forwards ray tracing is useful in sparse scenes, and also since it better represents the physical process of image acquisition (rays travel to the sensor in the real world). Backwards ray tracing is favoured for efficiency in this software.

Objects for the simulation can be either matrices of voxels or a polygon based mesh. In both cases, backwards ray tracing requires ray-polygon or ray-voxel intersection methods, which are often the bottleneck in the ray-tracing timeline. A brute-force method requires a ray to be checked with each polygon in the scene. In opaque scenes, surface normals can be used to overlook inaccessible polygons; however scene meshes can have thousands of vertices and faces. An octree implementation allows more efficient checking and indexing of intersections, as opposed to checking for intersections with all polygons for each ray. An octree discretises the scene into blocks, with each block either further discretised or containing mesh vertices. By implementing this discretisation, and using a tree-search algorithm, rays are checked against the octree branches until a node is hit, and the checked against the vertices contained within the node. This is invaluable when simulating large or detailed scenes.

4 Computational Approaches to Plenoptic Data

Rendering methods used on plenoptic data depend on the plenoptic architecture used. Approaches for both the original (Ng 2006) and focused plenoptic camera (Todor Georgiev & Lumsdaine 2010; T. Georgiev & Lumsdaine 2010) have been implemented in real-time to allow easy exploration of image data. Implementing rendering methods also allows for the development of post-processing algorithms. An example given is the depth map algorithm for the focused plenoptic camera, which exploits the multi-view stereo style set-up by using phase-correlation on neighbourhoods of microlenses to give the relative disparity for regions of space. Both the depth mapping and rendering of images is highly parallelisable, meaning vast increases in computational performance are achievable. The current performance of the MATLAB code to extract a depth map and surface render a 40 megapixel raw plenoptic image is between 20-60 seconds. The performance of the depth map algorithm used is dependent on how detailed a scene (or micro-image) is, with lack of image features leading to unreliable depth estimates.

The 3-D nature of a plenoptic camera allows us to extract the depth information of a scene, and the way in which the 4-D light field is parameterised allows for the use of computed tomography reconstruction techniques, as seen in medical image reconstruction. In terms of surface capture, a system can be simulated which utilises mirrors to provide multi-view analysis from a single acquisition. The reconstruction of this 3-D model is achieved iteratively through a maximum likelihood expectation maximisation (MLEM) reconstruction, which performs better than Radon transforms on multi-view data.

5 Conclusions

A toolbox for plenoptic simulation, rendering and data exploration has been developed which is capable of simulating millions of rays through a complex system of optics. The rendering of plenoptic images has many aspects which have been included in the software, on top of visualisations of surfaces and 3-D models. This toolbox will be useful in plenoptic camera design and, since any scene inputs are supplied by the user, many applications can be explored fully *in silico*. Future work includes incorporating Gaussian beam propagation and wave effects.

References

Georgiev, T. & Lumsdaine, a., 2010. Reducing Plenoptic Camera Artifacts. *Computer Graphics Forum*, 29(6), pp.1955–1968. Available at: http://doi.wiley.com/10.1111/j.1467-8659.2010.01662.x [Accessed June 17, 2013].

Georgiev, Todor & Lumsdaine, A., 2010. Focused Plenoptic Camera and Rendering. , pp.1–28.

Georgiev, T., Zheng, K. & Curless, B., 2006. Spatio-Angular Resolution Tradeoffs in Integral Photography. *Rendering …*. Available at: http://www.tgeorgiev.net/Spatioangular.pdf [Accessed October 18, 2013].

Lumsdaine, A. & Georgiev, T., 2008. Full resolution lightfield rendering. … *University and Adobe Systems, Tech. Rep*, (January), pp.1–12. Available at: http://www.tgeorgiev.net/FullResolution.pdf [Accessed June 17, 2013].

Ng, R., 2006. *Digital light field photography*. Available at: http://testcis.cis.rit.edu/~cnspci/references/dip/light_field_photography/ng2006.pdf [Accessed June 21, 2013].

Ng, R., Levoy, M. & Brédif, M., 2005. Light field photography with a hand-held plenoptic camera. *Computer Science …*, pp.1–11. Available at: http://www.soe.ucsc.edu/classes/cmps290b/Fall05/readings/lfcamera-150dpi.pdf [Accessed June 21, 2013].