Squeezing Video out of a Lytro Light Field Camera

Gerard Kennedy

U5185867

Supervised by

A/Prof. Jochen Trumpf

Mr. Sean O'Brien

Mr. Ben Nizette

October 2017

Australian National University

A thesis submitted in partial fulfillment of the degree of

Bachelor of Engineering

The Department of Engineering

Australian National University

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university. To the best of the author's knowledge, it contains no material previously published or written by another person, except where due reference is made in the text.

Gerard Kennedy
27 October 2017

Acknowledgements

Firstly, thank you to my primary supervisor, A/Prof. Jochen Trumpf. His patience, encouragement and support have made this project both an enjoyable and rewarding experience. I also appreciate the opportunity to undertake such an interesting project. Thank you also to my co-supervisors Sean O'Brien and Ben Nizette, for indulging my constant questions and providing such invaluable advice. Similarly to Alex Martin, thank you for providing such helpful insight and technical assistance.

I also wish to acknowledge and thank Jan Kučera for replying to a bombardment of emails, and providing his source code, without which some of my primary achievements would have been impossible. I would also like to acknowledge Donald Dansereau, Lucas Jirkovský and Montiel Abello for making their own software available, and for taking the time to help me to understand and implement it.

This work was only possible thanks to the assistance of all those mentioned here. I count myself truly lucky to have had the opportunity to work with such talented and generous people.

Also thank you to my girlfriend, Cass, for helping me edit this document, for being great, and for telling me that I had to include this line.

Abstract

Light field photography represents a new way of sensing the environment that provides higher dimensional information than standard cameras. While standard cameras record the intensity of all light rays that intersect their sensor, light field cameras also record the direction of these light rays, and the intensity associated with different directions. Many new applications of light field video are expected to be developed in the future, as this technology matures. This thesis aims to find new ways of using the cheapest light field camera, the Lytro Light Field Camera (called the 'Lytro' in this document), to develop and test applications of light field video.

A software package called 'Lytro Remote Shutter' has been developed, which builds on currently available software to provide a tool for capturing dynamic light field data sets. These data sets have associated pose information that is obtained by synchronising the Lytro with a Universal Robot 5 robot arm. This software is also capable of capturing a 'manually-triggered' video stream of $\sim$0.54fps by remotely triggering the camera in a timed loop. A 200 image data set has been captured using this software.

This thesis also presents a conceptual design of an Extraction System that has the potential to extract video from the Lytro. This System includes an 'Interceptor Printed Circuit Board' that can physically intercept light field data en route from the image sensor to the processor within the Lytro, and a USB-FPGA board containing a Field-Programmable Gate Array, to be used to process the data and transfer it to an external device. This thesis also presents a preliminary list of requirements that this system will need to meet if it is to successfully extract video from the Lytro.

Contents

List of Figures

List of Tables

Glossary of Terms

CBW	Command Block Wrapper
CCD	Charged-Coupled Device
CMOS	Complementary Metal–Oxide–Semiconductor
COACH	Camera On A CHip
DOF	Depth of Field
FPS	Frames Per Second
HiSPi	High-Speed Serial Pixel
MP	Mega Pixels
PCB	Printed Circuit Board
PPD	Parallel Pixel Data
RGB	Red-Green-Blue
SLAM	Simultaneous Localisation and Mapping
THz	Terahertz
TCP	Transmission Control Protocol

Chapter 1

Introduction

1.1 Overview

The *light field* is the full description of light as it travels in all directions through space around us. It can be thought of as being made up of infinitely many light ray vectors, each with a direction of propagation and a magnitude that is proportional to their intensity. As humans, we view the world by interpreting the intensity and frequency of the subset of light rays that intersect our eyes. However, our eyes, just like standard cameras, are incapable of recording the direction of these light rays. Recording the direction of these light rays allows for the light field to be fully defined at a given location.

Light field photography allows the direction of light rays to be captured. This means that a higher dimensional representation of light is captured by a light field camera compared to a standard camera. Light field cameras therefore provide a new, more detailed way of sensing light in an environment. This dense sensor information is already being applied to existing applications within the research fields of robotics and computer vision. Such applications include odometry, depth mapping, 3D image reconstruction, and image filtering. By capturing more information about a scene, these applications are able to gain more information from a single image. For example, 3D image reconstruction requires multiple images of a scene to be taken taken from different, known camera poses. This method is referred to as 'stereo vision'. The directional dimensions that are added by taking multiple standard images are captured in a single light field image.

Applications involving real-time light field capture are rare as light field video cameras are a new and relatively expensive technology. However, due to the similarities between light field vision and stereo vision systems, real-time applications of stereo vision such as robotic navigation, feature detection, and depth mapping are also expected to be easily translatable to light field video. There is, therefore, a large number of potential applications of light field video that await development and testing.

This thesis was undertaken with the aim of utilising or adapting the Lytro Light

Field Camera (hereafter called the 'Lytro') to allow it to be used to test and develop light field video applications. This was primarily done by attempting to extract video from the camera in the hope of providing the research community with a cheap light field video camera. While there are several commercial light field video cameras available, these are all too expensive to have seen wide spread adoption within the research community. This is one reason that very few applications of light field video have been developed.

This thesis presents a list of requirements and a conceptual design of a system that has the potential of extracting higher frequency video from the Lytro. This 'extraction system' includes an 'Interceptor Printed Circuit Board (PCB)' and a ZTEX USB-FPGA-Module 2.14 board containing an Xilinx Artix 7 Field-Programmable Gata Array (FPGA). It is hoped that this system can be used to intercept image data sent from the parallel interface of the Lytro's sensor, convert it to serial data to in turn be sent via the USB port on the USB-FPGA Module board to an external device. The design of this system is still conceptual and a more detailed design will need to be produced, along with software to run on the FPGA, before extraction will be possible. If this system is implemented successfully in the future it will represent another means of testing and developing light field video applications with the Lytro.

This thesis also presents software package, called 'Lytro Remote Shutter', capable of recording dynamic light field data sets and capturing 'manually-triggered' video of ∼0.54fps. A 'manually-triggered' video stream refers to video that is captured by triggering a camera as fast as possible, as opposed to a 'firmware-triggered' video stream, which would capture data continuously when initiated. This ∼0.54fps manually-triggered video stream is captured by wrapping a 'remote shutter' command into a timed loop. The capture of this video stream is implemented with a software package named the 'Lytro Remote Shutter', developed for this project.

This software package has been expanded to provide an interface between the Lytro and a Universal Robot 5 (UR5) robot arm, image downloading and deleting functionality, Lytro calibration, and light field image decoding. The software was created by using a range of other software packages, and is available online at [14]. The software can be used along with a UR5 to capture dynamic light field image data sets, such as the 200 image data set presented in this thesis. This software represents a new tool that can be used to test and develop light field video applications with the Lytro.

1.2 Document Structure

This thesis contains five chapters, the first of which is this introduction. Chapter 2, 'Background,' contains an overview of light field photography. This includes discussions of the physics involved (Section 2.1), its history (Section 2.2), its applications (Section 2.2.3), and a discussion of the motivations to test and develop light field video applications with the Lytro (Section 2.2.5). The Lytro is also introduced in this chapter, in Section 2.2.4.

Chapters 3 and 4 are each dedicated to one of the two broad approaches undertaken in the attempt to extract video from the Lytro. Approach **1**, which involved attempted video capture and extraction via software editing, is the subject of Chapter 3. This chapter presents an overview of the Lytro's communication protocols (Section 3.1), the Lytro Remote Shutter software package developed to capture a ∼0.54fps video stream and interface with a UR5 robot arm (Section 3.2), and method and results of capturing a dynamic light field data set using the software (Section 3.3).

Chapter 4 is dedicated to Approach **2**; physically intercepting the light field data before it reaches the Lytro's processor. This chapter includes summaries of the 'Riser PCB' (Section 4.4) and 'form tool' (Section 4.5) that have been manufactured to physically intercept image data within the Lytro. Also included are the results of the continuity testing (Section 4.2) and signal analysis (Section 4.6) that were performed in an effort to determine how best to physically extract the video stream. Finally, a list of requirements and conceptual design for an 'Extraction System' that has the potential to successfully extract video from the Lytro are presented in Sections 4.7.1) and 4.7.2 respectively.

Chapter 5, the conclusion, contains a summary of the key contributions and the future work that can be undertaken as a result. There are also a number of appendices that contain a variety of extra information. These include a set of images illustrating how to dissemble the Lytro (Appendix A), further information for, and examples of the Lytro's WiFi and USB communication protocols (Appendix B), the user manual for the Lytro Remote Shutter software package (Appendix C), key information about the interface of the image sensor within the Lytro (Appendix D), a detailed overview of the design of the Riser PCB (Appendix E), an overview of the construction of the form tool (Appendix F), more information about the ZTEX USB-FPGA-Module 2.14 board (Appendix G), and, finally, a detailed overview of the design of the Interceptor PCB (Appendix H).

1.3 Summary of Contributions

- The conceptual design of a system that has the potential to be used to extract light field video from the Lytro. The design consists of an Interceptor PCB and a ZTEX USB-FPGA-Module 2.14 board containing an Xilinx Artix 7 FPGA.
- A list of requirements that will need to be met for successful extraction of light field video from the Lytro.
- The Lytro Remote Shutter software package that will interface the Lytro with a UR5 robot arm and capture a ∼0.54fps video stream. The software will also capture images with keyboard trigger, perform camera calibration, and decode of light field images. The software allows for light field images to be captured in synchronisation with a UR5, thus providing camera pose information as ground truth.
- A 200 photo dynamic light field data set with associated camera pose information.
- A summary of the interface between the Lytro's image sensor and image processor, which includes the physical location of the light field data signals and an overview of the protocols and data encoding used to transmit and receive these signals.
- A summary of the Lytro's USB and WiFi-based communication protocols.
- A 'Delete Photo' command for the Lytro's WiFi, TCP-based protocol.
- A Riser PCB that can be used to physically intercept signals passing between the image sensor and image processor within the Lytro.

Chapter 2

Background

The following chapter is split into two sections, which together present the theoretical context and motivation for this project. Section 2.1 outlines the theory behind light field cameras, including what the light field is, how it is captured, how it is rendered, and how light field photography differs from standard photography. The theory in this section is mainly sourced from [2, 15, 16, 17, 18, 19, 20]. Section 2.2 presents the scope of the project, including the historical context, and the advantages and applications of light field photography. A summary of how this project fits into the existing body of knowledge is also provided. Section 2.2 is primarily sourced from [2, 3, 16, 17, 20, 21, 22]. The Lytro camera is also introduced in this chapter in Section 2.2.4.

2.1 Theory

2.1.1 Light

Visible light propagates through space as part of the electromagnetic spectrum. This part of the electromagnetic spectrum incorporates frequencies of between 430 and 700 Terahertz (THz), and contains within it all the colours that we see. These colours are visible to us due to the different frequencies at which light is reflected and absorbed by different objects. Light can also be deflected when it passes through an object, in a phenomenon known as refraction. It is refraction that allows light to be focused by a lens onto an image sensor.

2.1.2 The Image Sensor

Image sensors come in two main varieties: Complementary Metal Oxide Semiconductor (CMOS), or the older Charge-Coupled Devices (CCD). When the image sensor is exposed, photons hit the photoactive areas on the sensor, called 'pixels', and an electrical charge is produced. It is the differing ways in which these charges are produced that differentiates CMOS and CCD sensors. Placing a filter, such as a Bayer filter, over the pixels on a sensor allows the frequency of light to dictate the amount of electric charge produced. Pixels can therefore be used to determine

the intensity of different colours within a scene. Intensity is often also referred to as 'irradiance' and is a measure of power per unit area. These filters comprise of patterns of colour filters that allow different wavelengths of light through (see Figure 2.1). The missing information between consecutive colour filters of the same type is added to the scene via interpolation to create the final image.

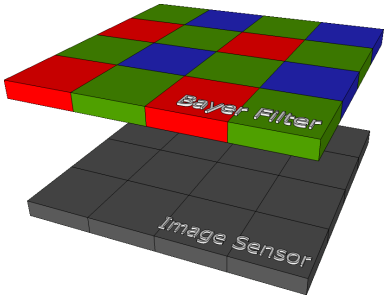


Figure 2.1: Bayer filter above image sensor pixels [1] (best viewed in colour).

2.1.3 Cameras

In a standard camera, light is focused by a lens, through a filter, onto an image sensor. When this sensor is exposed the pixel responses are integrated over the exposure duration. The image that is produced has a single plane of focus that is dictated by the distance between the lens and the sensor (the focal length). Parts of the scene that are not within the plane of focus are recorded by the sensor as unfocused, blurry spots. This is because light rays propagating from these depths are not precisely focused on to the sensor by the lens. However, capturing the full light field, rather than just the summation of photon responses, allows for light rays at all depths to be focused on to the sensor. The remainder of this section is dedicated to explaining what the light field is, how it is captured, and the advantages of capturing it.

2.1.4 The Light Field

While the true behaviour of light is complicated, its movement through space is usually simplified to a light ray vector for image-based applications. This simplification allows the direction and intensity of light rays to be conserved while the more complicated properties of light are ignored. Each light ray is the result of reflection of light by a body, and contains within it a projection of that body. This projection can be captured by a camera at a given location. For example, a pin-hole camera works by capturing a small subset of these rays, and will capture a different subset (and thus a different perspective) depending on its location relative to the body (see

Figure 2.2). It is therefore not only the intensity of light rays, but also their orientation and direction (i.e. their angle of incidence) that dictates the resultant image. It follows that to fully describe the behaviour of light as it flows through a camera lens to the image sensor, one must quantify not only the intensity of the light rays in this volume, but also their orientation and direction. This full description of light rays is what is known as the *light field*. The light field can therefore be thought of as the infinite collection of light ray vectors, each with a magnitude proportional to their intensity, that exists around us [23]. The question that arises at this point is, how many dimensions are required to fully define light ray vectors?

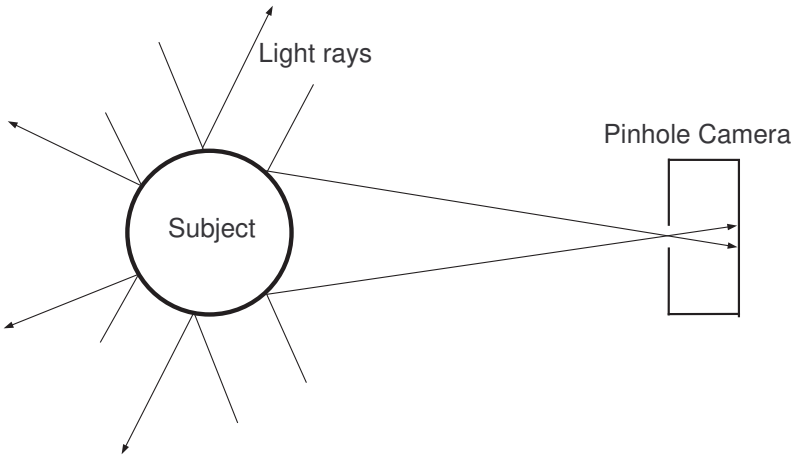


Figure 2.2: Light rays reflecting off a body and being captured by a pin-hole camera.

In 1991 Adelson and Bergen were the first to provide the answer to this question with their *Plenoptic Function*, which has a 7-dimensional (7D) input [23]. This function, derived from the Latin word *plenum* meaning 'full', contains 3 spatial coordinates, 2 directional coordinates, 1 time coordinate and 1 frequency coordinate [19]. These 7 dimensions fully define a light field. However, not all 7 dimensions need to be captured to fully define a light field for light field photography applications. For example, it can be seen immediately that time is unnecessary for static images. Frequency is also not required, as it can be replaced with the 3 channels of a colour space implemented with a filter such as a Bayer filter [19]. Finally, one of the spatial coordinates is unnecessary, as the nature of a light ray does not change along the axis of propagation unless it is blocked [24]. We are therefore left with a 4D subset of the Plenoptic function that fully describes the light field for light field photography applications.

2.1.5 The Light Field Camera

To take a standard photograph the intensity of the light field is recorded at a pixel by integrating over all individual light ray contributions at that point. A standard image therefore represents a 2D array of values, each corresponding to the intensity of the light field at that location. However, to capture the full light field at each location, rather than just the intensity, the direction of the light rays intersecting that point must also be recorded.

A variety of different methods have been proposed and implemented to capture this 4D light field. A comprehensive list of typical methods, with links to the corresponding literature, is available in [20]. The four key areas within which these methods can be categorised are: 'multi-sensor', 'time sequential', 'frequency multiplexing', and 'spatial multiplexing' [20]. Of these four methods, spatial multiplexing is dealt with here, as the Lytro camera falls into this category. Spatial multiplexing involves capturing "an interlaced array of elemental images representing samples from different 2D slices of the light field" [20]. These 2D slices are the micro-images discussed below in Section 2.1.7. Four key approaches to spatial multiplexing, along with a list of corresponding cameras within, is provided in [17]. These approaches can be summarised as follows: **1.** a main lens focusing onto an array of micro-lenses over an array of detectors of the same dimensions, **2.** a micro-lens array over a photo sensor, **3.** aspherical lenses used to create depth-specific blur, and **4.** a main lens focusing on to an array of micro-lenses over an image sensor [17]. These different approaches are discussed furthin within Section 2.2.

The different ways that these four spatial multiplexing cameras can capture a light field image may still appear confusing at this point. To provide greater clarity, Approach **4**, which is the approach utilised within the Lytro, will be analysed in detail in Section 2.1.6. The optics discussed in relation to this approach will hopefully allow the reader to understand how the other approaches would function.

2.1.6 The Micro-lens Array

The Lytro employs an array of ∼130,000 hexagonally arranged micro-lenses placed ∼25μm (one focal length) from the image sensor [11]. Light rays are initially focused by the camera's main lens, just as is the case in a regular camera. However, instead of the light rays then striking the image sensor, they are re-focused on to the image sensor by the micro-lens array [17]. Micro-lenses can be thought of as the pixels in a standard camera, while pixels in a light field camera are used to determine light ray direction [25], as discussed below. A diagram showing several light rays propagating

through a Lytro lens system is provided in Figure 2.3. Note that the Bayer filter is omitted from Figure 2.3 and further figures. The filter can be thought of as simply being part of the image sensor.

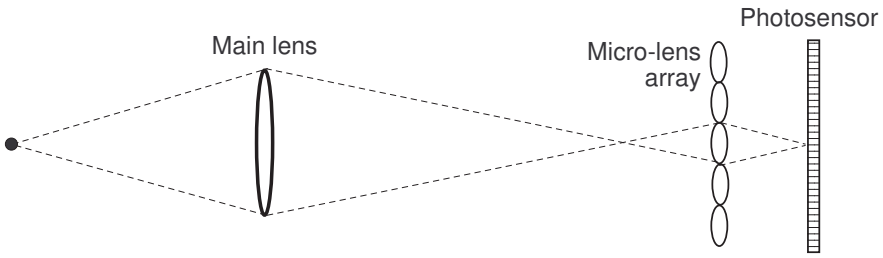


Figure 2.3: The Lytro's lens configuration.

For this configuration to be capable of capturing the light field within the camera lens, all 4 dimensions of the reduced Plenoptic function must be captured. The two-plane parameterisation that is used to accomplish this can be expressed as

$$P = L(x, y, u, v) \tag{2.1}$$

where P is the Plenoptic Function, L is the light field, and (x, y, u, v) are spacial coordinates, with (x, y) representing the sensor plane, and (u, v) representing the micro-lens plane. The relationship of the spatial coordinates is illustrated in Figure 2.4.

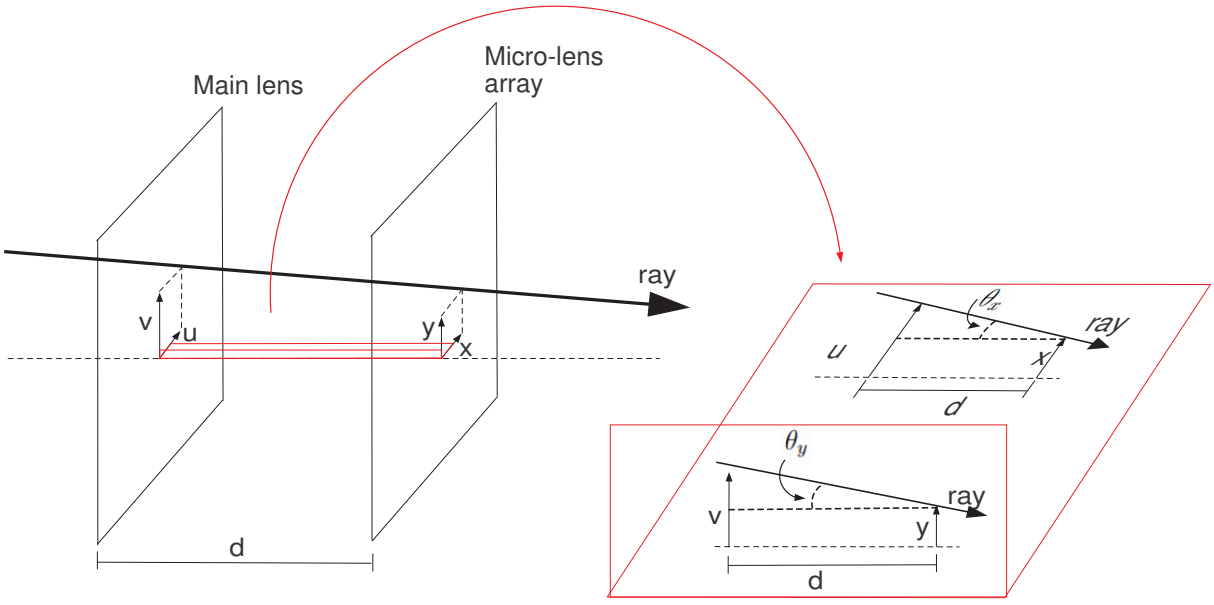


Figure 2.4: Two-plane plenoptic parameterisation. The same ray is represented from 3 different angles to illustrate the angles θ_x and θ_y. Based on [2] (best viewed in colour).

The orientation and direction of a light ray can be characterised by its x and y coordinate, and its direction (θ_x and θ_y). θ_x and θ_y can be calculated using the distance (d) between the lens plane and the sensor plane, and the relationships [2]

$$\theta_x = \tan^{-1}\left(\frac{x-u}{d}\right), \tag{2.2}$$

$$\text{and } \theta_y = \tan^{-1}\left(\frac{y-v}{d}\right). \tag{2.3}$$

While this clarifies how the two-plane parametrisation can be used to obtain the direction and orientation of a light ray, it may still be unclear how these values (x, y, u, v) are measured in the first place. Excellent discussions of how the (x, y, u, v) values are recorded in lens-array based light field cameras can be found in [2, 16]. These discussions are summarised here.

To measure the angle of incidence of light rays, it must be remembered that there are many pixels underneath each micro-lens. Using a method that is similar to Single Lens Stereo (see [16] for a description of this technique), the angle of incidence can be determined by tracking which individual pixels have been reached by each light ray. Figure 2.5 provides an illustration of how different combinations of pixel responses indicate different locations of the subject, by displaying two such locations.

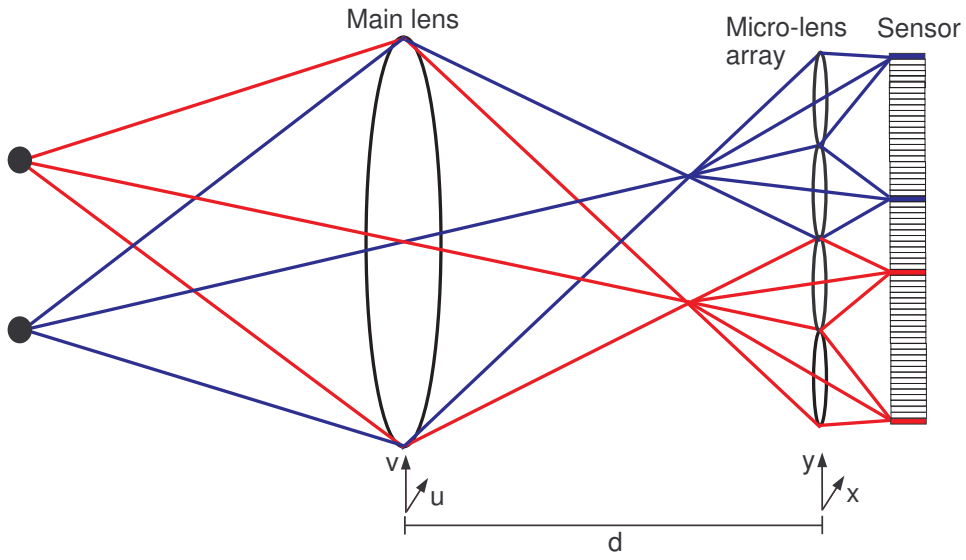


Figure 2.5: Correlation between subject location and pixel response. Light propagating from different points will reach different combinations of pixels under the micro-lenses. The parametrisation $L(x, y, u, v)$ is also shown (best viewed in colour).

Using the location of the pixels struck by a light ray, the corresponding micro-lens

(the x, y coordinates) can be determined, as can the corresponding location on the main lens (the u, v coordinates) [17]. From these coordinates, the direction of the light ray can be determined via Equations (2.2) and (2.3) [2]. The location of pixel responses, combined with the relationships defined by Equations (2.2) and (2.3) therefore provides all the information required for the full $L(x, y, u, v)$ parameterisation, thus allowing light field images to be captured by the Lytro.

More information regarding the $L(x, y, u, v)$ parameterisation can be found in [2, 16, 17, 18, 19].

2.1.7 Light Field Image Rendering

Finally, we must examine how light field data is arranged to produce comprehensible images. The micro-lens array in light field cameras causes an array of 'micro-images' of the scene to be captured, one beneath each micro-lens. These micro-images can be transformed to 'macro-images' (standard images of the full scene) by summing all the pixels beneath each micro-lens [2]. The micro-images are all slightly different due to their location relative to the main lens. This difference illustrates the capture of the extra, spacial coordinates by the micro-lens based light field camera.

By choosing only pixels that correspond to a certain direction within the main lens aperture, the resultant macro-image will appear to have been captured from that direction [2, 18]. For example, selecting the centre pixel beneath each micro-lens will produce the centre macro-image. Likewise, the depth of field can be varied by summing over a wider or thinner band of pixels beneath each micro-lens [2].

It is hoped that the overview of light field photography provided in this section is sufficient for the reader to understand how the light field is captured by the Lytro. More in-depth explanations of the optics discussed here can be found at [2, 16, 17, 18, 19].

2.2 Project Scope

This section is a summary of the context within which producing new ways of using the Lytro to test and develop light field video applications is valuable. This section presents an overview of the historical development of light field photography, from its beginning in 1908 to the current commercially available cameras. It also presents the advantages and applications of light field photography and video.

2.2.1 Historical Context

"Is it possible to create a photographic print in such a manner that it represents the exterior world framed, in appearance, between the boundaries of the print, as if those boundaries were that of a window opened on reality?" - Gabriel Lippmann [26]

In 1908 Lippmann introduced the world to the concept of light field photography in the form of his 'Integral Camera' [26]. Lippmann proposed a camera comprised of an array of tiny lenses that could be used to project a series of microscopic images of a scene onto film (see Figure 2.6). This film could then be back-lit allowing for the scene with its true size and depth to be viewed without the need for a stereoscope, which had been invented five years previously [19, 26]. This method is the same as Approach **2** discussed in Section 2.1.5, and has the drawback of not capturing the spatial coordinates of the Plenoptic Function described in Section 2.1.4.

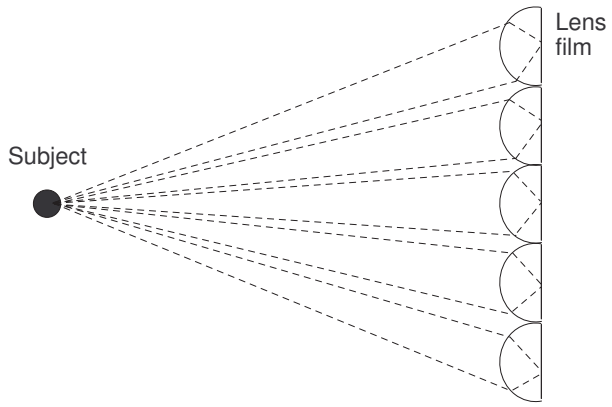


Figure 2.6: Lippmann's 'lens film'.

Throughout the 1900s and early 2000s a variety of integral cameras were designed, summaries of which can be found in [17, 18, 20, 21]. An active area of research throughout this time involved optimising light field photography for a range of different applications. In the case of the spatial multiplexing light field cameras, this has been done using variations of the four Approaches discussed in Section 2.1.5. For example, Approach **1** involves a main lens focusing light on to a micro-lens array and then an image sensor, however, with only one pixel under each micro-lens, its resolution is severely limited [17]. This approach was developed in 1983 as a means of increasing the effective photo-diode aperture of a CCD sensor, rather than to create an image, which was the focus of subsequent approaches [27]. Approach **2**, which was the variation Lippmann introduced in 1908, and was used again in 1994 as a means of studying the application of insect eye optics to photography [28]. This approach involves a micro-lens array over a photo-diode array, and in 1994 was used

to produce an image of 16 × 16 pixels and 8 shades of grayscale [28]. However, due to the lack of a larger main lens, this method does not have the necessary focusing capability to produce high resolution images [17]. Approach **3** involves combining a micro-lens array with wavefront coding techniques [29]. This method is similar to that which is implemented within the Lytro, although wavefront coding allows for only one reconstructed image, rather than a range within the lens aperture [17, 29]. Approach **4** involves a main lens focusing on a micro-lens array, and was proposed by Adelson and Wang in 1992 as a means of inferring object depth with a single camera [16, 17].

Adelson and Wang presented a light field camera that could be created by augmenting a standard camera with a micro-lens array, placed exactly one focal length from the image sensor [16, 18]. Adelson and Wang's design, which was created to allow for the capture of vertical and horizontal parallax in the same image, was adapted by Ren Ng in his 2006 PhD dissertation, resulting in a portable camera [17, 18]. After his dissertation Ng founded the company Lytro, Inc., which produced the Lytro camera six years later [2]. More information regarding the development of light field technology during this era can be found in [18, 19]

Another company, Raytrix, released their R11 light field camera at a similar time [2]. The R11 is capable of capturing images containing 3MP and 10.7 Megarays. It is also capable of video up to 10fps [30]. Raytrix and Lytro, Inc. employ two slightly different variations of Approach **4.**: Lytro, Inc. place the focal place of the main lens on the main lens axis, while Raytrix's design does not. The pro's and con's of these two variations (mainly revolve around depth of field and) are discussed in greater detail in [25]. More recent commercial light field cameras include Lytro Inc.'s Illum (4MP, 40 Megarays [31]) and Raytrix's R5 (1MP, 4.2 Megarays, 15 frames per second (fps) [32]) and the R29 (7.25MP, 29 Megarays, 5fps [33]). While Lytro, Inc. and Raytrix remain the only two manufacturers of light field cameras as complete, portable units, a more recent proposal is to implement light field cameras using modules containing tiny micro-cameras [34].

One company proposing such technology is Pelican Imaging, who's module is envisioned to be capable of 8MP images and 1080p, 30fps video, while also being small enough to fit inside smart devices [34]. Other companies becoming involved in light field photography include Toshiba, who are currently also working on similar modules to Pelican Imaging, and Apple, who recently purchased Linx, a company similar to Pelican Imaging [35, 36].

2.2.2 Advantages of Light Field Photography

Standard photography limits the user to a pre-determined plane of focus. As the light field captures the full light field within the camera, all the light rays that pass through main lens are focused onto the image sensor. As all light rays are focused, it is possible to choose the depth of field (DOF), and the plane of focus after the image is captured [18]. This removes the need to pre-focus up to a limit [18].

A second, less obvious benefit is the de-coupling of the relationship between the DOF and aperture. While standard cameras are capable of capturing an image that is almost entirely in focus by using a very small aperture, such as f/22 [17], light field cameras are capable of the same DOF with an aperture as large as f/4 [18]. This in turn means that shorter exposure times are required to capture images with a large DOF, thus reducing noise and the possibility of motion blur [18].

Finally, the ability to produce images that relate to any region of the main lens aperture means that computer vision techniques that require images of an object from different angles (i.e. techniques that require parallax) can be implemented using single light field photos [20].

These advantages have allowed the light field camera to be have wide-ranging current and potential applications, particularly in the research fields of robotics and computer vision. These applications are discussed in Section 2.2.3.

2.2.3 Applications of Light Field Photography

The de-coupling between DOF and aperture that is provided by light field photography has wide ranging potential applications. There are many situations in which a large DOF is required, but the traditional method of achieving this (decreasing the aperture) is impossible. Decreasing the aperture requires an increase in exposure duration to maintain the illumination in the resultant photograph. Any situation that requires a clear image in low or difficult lighting conditions, and/or while either the camera or subject is moving, will benefit from light field photography. Such situations include many computer vision algorithms, that often require use in the presence of occluders such as fog, snow or water) [37], microscopic imaging [29], sport photography, and movie filming.

The potential for the use of light field photography in the research fields of robotics and computer vision can, in turn, be broken down into several areas. Due to its higher dimensional representation of light, light field photography is expec-

ted to lead to new developments in applications including odometry [38], depth mapping [20, 39, 40], image segmentation [20, 41], 3D reconstruction [42, 43], image filtering [44, 45] and Simultaneous Localisation and Mapping (SLAM) [21, 46]. For example, within the research area of filtering, there is potential for light field cameras to be used to capture dense measurements that can be used to implement infinite-dimensional, non-linear observers [44]. An observer is a filter that estimates the state (some set of variables governed by the system) of a dynamic system using the different between measured and predicted outputs [44]. This particular application is a focus for the Research School of Engineering at the Australian National University, where this project has been undertaken. The link between this thesis and this research interest is discussed in more detail within Section 3.3.

With several of the more recent light field cameras becoming capable of recording video, there has been a corresponding shift in interest within the academic literature towards real-time, dynamic light field capture. Examples of this literature include a 2011 paper by D. Dansereau *et al.* [38] and a 2013 paper by F. Dong *et al.* [21]. These papers provide some of the first examples of the concept of light field video being utilised for robotics applications, specifically visual odometry and SLAM. A key problem that needs to be overcome for light field video to be utilised for such applications is the required processing speed. The scope of this problem can be grasped by studying the Lytro. The Lytro produces images containing 11 'Megarays' at a resolution of 12 bits [11]. 'Megarays' refers to the number of megapixels (MP) in the image sensor, but is expressed differently to indicate the existence of the micro-lens array [47]. Therefore a video capture rate of 30fps would lead to a data rate of: $(30\text{fps}) \times (11 \times 10^6\text{MP}) \times (12\text{bits})/(8\text{bits/byte}) = 0.495\text{GB/s}$. Dansereau and Dong take different approaches to overcome this issue. Dansereau introduces three distinct closed-form solutions [21], while Dong studies ways to reduce data while maintaining accuracy [21]. The data reduction achieved by Dong comes from a combination of reducing motion speed, sensor resolution and field of view [21].

While there are still relatively few examples of literature that investigate light field video, it is important to remember that light field cameras are very similar to stereo camera systems. Stereo camera systems are essentially larger light field cameras that contain an image sensor for each lens. Applications of real-time stereo vision are therefore likely to be easily transferable to light field cameras [16]. Potentially relevant applications of real-time stereo vision include (but are by no means limited to) mobile robotic navigation [48, 49, 50], feature detection [51, 52], and depth mapping [53, 54, 55].

Light field cameras are an imaging system with wide ranging current and potential applications. Light field cameras have only recently become commercially available, and it appears likely that they will prove to be a valuable vision system.

2.2.4 The Lytro

The camera that forms the basis of this thesis is the Lytro, shown below in Figure 2.7.

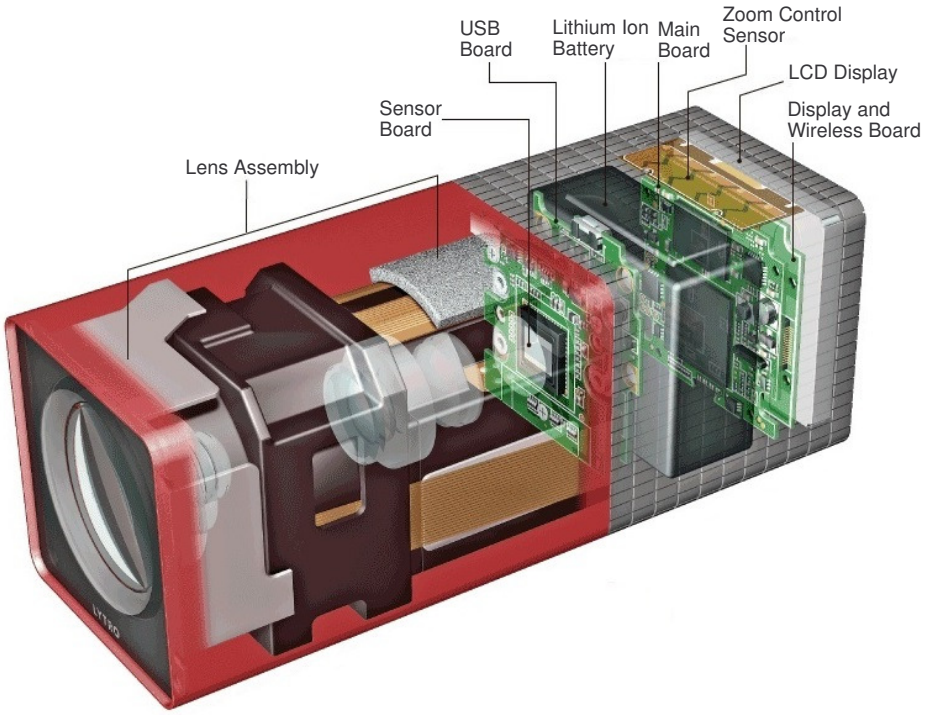


Figure 2.7: The Lytro [3] (best viewed in colour). The names of the three Printed Circuit Boards: the 'USB board', 'main board', and 'sensor board', are used extensively throughout this document.

This camera contains an array of ∼130,000 hexagonally arranged micro-lenses. These lenses are situated ∼25μm (one focal length) from the image sensor, which is located on the sensor board as shown in Figure 2.7 on page 16. The image sensor is an Aptina MT9F002 14.4 Mpx 1/2.3" CMOS sensor, with a cropped output frame containing 10.7Mpx [11]. The CMOS implements a standard colour Bayer filter array (see Figure 2.1 on page 6) to capture colour information [11]. Data from the sensor passes from the sensor board to the main board where it is processed by a 12th generation Camera-on-a-Chip (COACH) manufactured by Zoran [11]. To utilise this particular processor in their camera, 1GB of SK Hynix DDR2 SDRAM memory, a 32-bit RISC digital image processor and a 1.52" touch-enabled, 128×128

pixel LCD display are also used [56].

It is the inclusion of the LCD display that provides the most promising indication that video may be extracted from the camera. When turned on, the LCD displays a live feed of the image sensor data. In other words, a video stream is already being captured by the camera. This project focuses on extracting this video stream. The video is produced as a result of the rolling shutter functionality of the CMOS sensor. The rolling shutter mode is the default operation mode of the CMOS sensor, and involves streaming image data continuously as it is captured by rolling the shutter across the sensor [5].

As stated in Section 2.1.5, the Lytro implements Approach **4** of the typical methods of implementing spatial multiplexing light field photography.

In-depth summaries of the internal components of the Lytro can be found in its user manual [22], the Light Field Forum [3] and a website [11] and Masters thesis by Jan Kučera [2]. Data sheets for the COACH processor and CMOS sensor are referenced repeatedly throughout this document, and can be found at [7] and [5] respectively. An outline of how to dissemble the Lytro, with photographic illustration is also provided in Appendix A.

2.2.5 Gap in the Field

While the previous decade has seen a shift in light field literature towards real-time light field data, there are still relatively few applications being tested and developed [21]. One reason for this is that current light field cameras are either incapable of video capture or too expensive for wide spread academic use (see Table 2.1). This issue will be rectified when light field video cameras become available on integrated circuits, however this technology is yet to reach the market, as discussed in Section 2.2.1.

Table 2.1: Light field camera price comparison.

Model	**Approximate Price (AUD)**	**Capable of Video**	**Source**
Lytro	180	N	[57]
Lytro Illum	1,400	N	[57]
Raytrix R5	7,000	Y (15fps)	[2]
Raytrix R11	42,000	Y (10fps)	[30]
Raytrix R29	49,000	Y (10fps)	[58]

Light field video is expected to influence a large range of research areas within the fields of both robotics and computer vision as it presents a revolutionary way of sensing the environment (as discussed in Section 2.2.3). This project aims to find and develop new ways of using the Lytro to test and advance applications of light field video. This is primarily done by attempting to extract video from the camera. This project therefore represents the first documented attempt to produce cheap light field video for academic research purposes. The resultant conceptual design presented in this thesis has the potential to produce such a camera in the future. This would allow for applications such as those discussed in Section 2.2.3 to be cheaply studied and developed.

Aside from the extraction of video from the Lytro, the other main focus of this thesis is to capture a slow, manually-triggered video stream and associate the corresponding images with (orientation and direction) pose information . This is undertaken as it represents an alternative way of using the Lytro to test and develop applications of light field video. Applications involving dynamic (moving camera) light field video require the relative pose of the camera to be known between consecutive frames. Such applications can be tested with Lytro using the method and software presented in this thesis. The software package that is developed to capture these images and determine the corresponding camera pose represents the first documented instance of a tool that allows the Lytro to be used to develop dynamic light field video applications.

Chapter 3

Software Implementation

The following chapter describes the results of attempts to extract video from the Lytro via a software based approach. The Lytro's WiFi and USB communication protocols are analysed and summarised in Section 3.1. This analysis led to the identification of a new 'delete photo' command, but also to the conclusion that using these protocols to re-route and extract a video stream would be unfeasible. Instead, a software package called 'Lytro Remote Shutter' has been developed to utilise the protocols. This software package is capable of capturing a ∼0.54fps video stream and interfacing with a Universal Robot 5 (UR5) robot arm to capture dynamic light field data sets. This software is presented in Section 3.2. The steps required to capture data sets and obtain camera pose information are presented in Section 3.3, as are the results of capturing one such data set. Finally, the remaining problems associated with capturing data sets and obtaining the pose information, along with possible causes and potential fixes, are outlined in Section 3.4.

3.1 Communication Protocols

The transport layer of the Lytro's USB and WiFi communication protocols are presented in this section. The transport layer is the fourth layer in the OSI model, and provides functionality to transfer variable length sequences of data. Software such as the Lytro, Inc. Desktop software [59], Lytro Compatible Communicator [11], and Lyli [60] communicate with the Lytro via these protocols.

The Lytro uses its own adaption of the USB Mass Storage (USBMS) 0x05 sub-class, which is an obsolete protocol for communicating with USB floppy drives [9]. This protocol sends commands by wrapping them in a Command Block Wrapper (CBW), as illustrated in Table 3.1. Command Block Wrapper is a packet containing a command block and associated information [10]. A description of each part of the CBW, and an example Lytro USB message are provided in Appendix B.

Table 3.1: USBMS Command Block Wrapper [9, 10]. Protocol described in Appendix B.

bit → **Byte ↓**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0-3	dCBWSignature							
4-7	dCBMTag							
8-11	dCBWDataTransferLength							
12	bmCBWFlags							
13	Reserved(0)						bCBWLUN	
14	Reserved(0)				bCBWCBLength			
15-30	CBCBW							

Communication with the camera over WiFi is implemented using an adaption of the Transmission Control Protocol (TCP), as illustrated in Table 3.2. A description of each part of the protocol, and an example Lytro WiFi message are provided in Appendix B.

Table 3.2: Lytro TCP protocol [11, 12]. Protocol described in Appendix B.

bit → **Byte ↓**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0-4	Source Port								Destination Port							
4-8	Sequence Number															
8-12	Acknowledgement Number															
12-14	Data Offset				Reserved			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN
14-18	Window Size								Checksum							
18-20	Urgent Pointer															
20-24	Signature															
24-28	Content/Buffer Length															
28-32	Flags															
32-34	Command															
34-48	Parameters															
48-62	Optional Payload															

Full lists of the known USB and WiFi transport layer protocol commands can be found at [11] and [9] respectively. The only command identified during this project

that is not included in these lists is a 'Delete Photo' command. This command was identified in collaboration with Jan Kučera, and is used in the software package described in Section 3.2. This command is sent via the protocol shown in Table 3.2, with 'Command' = C0 00 05 00, and a 24 byte payload that is shown in Table 3.3. A full description of the known USBMS commands is available at [9], and a full description of the known WiFi based TCP commands is available at [11].

Table 3.3: 'Delete' command payload [13].

Offset	**Size**	**Type**	**Contents**
0x00	8	string	folder name postfix, right padded with zeros
0x08	8	string	file name prefix, right padded with zeros
0x10	4	int	folder number
0x14	4	int	file number

The fact that the 'Delete Photo' command is the only new command presented in this thesis is not unexpected. The Lytro does not use standard protocol libraries, and commands that are not already implemented by the official Lytro Desktop software [59] can therefore only be found by guessing. Attempts to guess new commands based on the structure of known commands proved unsuccessful as none of the accompanying parameters are known.

3.1.1 Manually-Triggered Video

A 'manually-triggered' video stream involves triggering the Lytro's shutter as quickly as possible, as opposed to a 'firmware-triggered' video stream, which would capture data continuously when initiated. The limiting factors involved in capturing such a stream include the frequency at which the Lytro can capture images and clear them from its internal memory, and the maximum data rate at which images can be extracted from the camera. This second requirement is important if the video is required on a remote device in real-time.

The result of rapidly manually triggering the Lytro is displayed in Figure 3.1 on page 22. This figure indicates an average time between photos of ∼1.86 seconds, which corresponds to a video stream with a frame rate of ∼0.54fps. This slow frame rate is due to the time required to process a light field photo and clear it from the Lytro's internal memory. The fastest 'manually-triggered' video stream that can be captured by the Lytro is therefore ∼0.54fps.

The reason that the Lytro can display 60fps video on its LCD screen, even though

a single photo takes an average of 1.86 seconds to capture and process, is because of the two modes in which the CMOS is capable of capturing images. In 'global reset' mode all pixels are exposed and processed at once, while in 'rolling shutter' mode the pixels are exposed and the data processed in a constant cycle, allowing for a higher frame rate [5]. The Lytro therefore appears to take photos in global reset mode, while the LCD display is the result of the rolling shutter mode. These two sensor modes also provide the key difference between 'manual-triggering' and 'firmware-triggering'. Manual-triggering utilises the global reset, while firmware triggering utilises the rolling shutter.

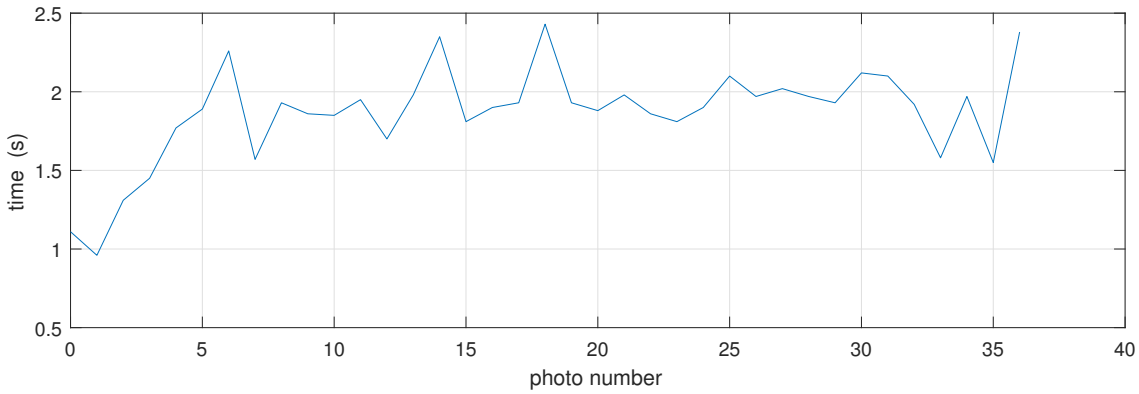


Figure 3.1: Time between images when Lytro is triggered rapidly.

While ∼0.54fps is a relatively low frequency video stream, such a stream can still be used to test and develop applications of light field video. For example, it can be used to capture dynamic light field data sets by associating camera pose information with each photo (as discussed in Section 3.3).

The Lytro Compatible Communicator software package already includes remote-triggering functionality [11]. The corresponding command has been wrapped into a loop, which has been slowed down to accommodate the required frame rate. This has resulted in a ∼0.54fps manually-triggered video stream being captured by the Lytro. The software used to capture this stream is included in the Lytro Remote Shutter software package presented in Section 3.2.

Real-time extraction of this video from the Lytro is also desired, as this would allow real-time processing of light field images to occur. However, despite the maximum data rate of the Lytro's WiFi chip (150Mb/s [61]) theoretically allowing for a transfer rate of up to∼1.2fps, this rate was found to be less than 0.1fps when tested with the Lytro Compatible Communicator [11]. Furthermore, when the Lytro connects to a remote computer via USB it changes to a 'charging' state that does not allow for

photos to be taken. The state diagram shown in Figure 3.2 has been constructed to illustrate these state transitions. This change means that it is impossible to capture a video stream and simultaneously extract data via USB.

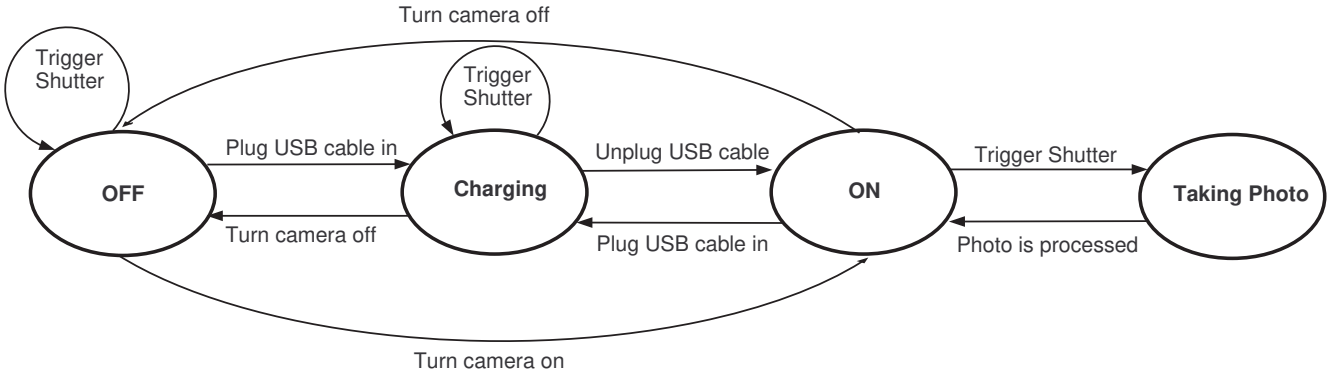


Figure 3.2: Lytro state machine diagram.

As no command has been found that will either cause the camera to trigger while charging or cause the camera to change state, it appears unfeasible to extract real-time image data via the USB interface. Therefore, while a manually-triggered video stream has been produced, a means of extracting this stream from the camera in real-time has not been found.

3.2 Lytro Remote Shutter Software Package

The Lytro Remote Shutter software package is primarily designed to be an interface between the Lytro and a UR5. This allows for the capture of dynamic light field data sets. The method of capturing data sets is discussed in Section 3.3. The capture of the data sets requires synchronisation with a UR5 so that the Lytro's pose can be determined relative to a known reference frame. The interface between the Lytro and the UR5 involves a number of stages that are discussed in full within the user manual (Appendix C). Briefly, these stages include calibrating the Lytro, communicating a desired trajectory to the UR5, triggering the Lytro automatically between UR5 movements, extracting the data set to a remote PC when finished, deleting images from the camera (to allow consecutive data sets to be captured), decoding the images, and estimating the position of the Lytro relative to a checkerboard.

The distance between the optical centre of the Lytro (estimated relative to a checkerboard) and the position of the tool point of the UR5 (provided to the UR5 by the

software) can be calculated. This calculation provides the location of the Lytro's optical centre with respect to a fixed reference frame, and is a focus of Section 3.3. The software also provides two other methods of triggering the Lytro; using `SPACEBAR`, and using a timed loop. The time-loop captures the ∼0.54fps video discuseed in Section 3.1.1).

Lytro Remote Shutter utilises the `Shutter Trigger` function of the Lytro Compatibility Communicator [11] to trigger the Lytro's shutter. Lytro Remote Shutter also implements the Lytro Compatibility Communicator's [11] WiFi interface to communicate with the Lytro, and uses other functions that allow for photo downloading and transferring camera information. The code required to perform these functions was kindly provided by Jan Kučera in the form of a Dynamically Linked Library. The Matlab Light Field Toolbox v0.4 [62] is also used to perform the calibration, position estimation, and decoding functionality of Lytro Remote Shutter. All sources used to create this software package are fully acknowledged in user manual. The Lytro Remote Shutter software package is available online at [14].

3.3 Dynamic Light Field Data Set

The Lytro Remote Shutter software package has been used to capture a 200 image data set of a checkerboard cube. The cube was chosen to aid in efforts to develop the observer mentioned in Section 2.2.3. The process by which this software can be used to capture the data set and associated pose information is outlined below.

The Lytro is attached to a UR5 robot arm, which allows its position to be tracked accurately. The Lytro is attached in such a way that its longitudinal axis is oriented as close to perpendicular to the plane of the tool point of the UR5 as possible. This corresponds to the Z_1-axis of the tool point (see Figure 3.3 on page 25). For the purpose of calculating the pose information, it is assumed that the optical centre of the Lytro lies exactly on the Z_1 axis.

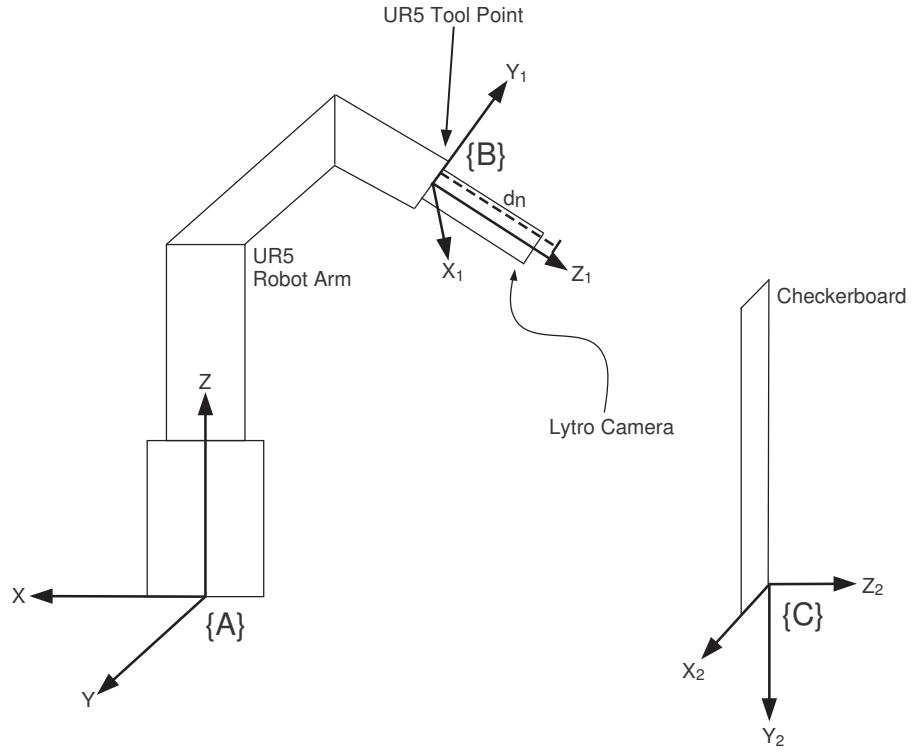


Figure 3.3: Lytro-UR5 robot arm setup illustrating the three reference frames {A}, {B}, and {C}, and the distance 'd_n' at pose n between {A} and {B}.

The setup contains three reference frames. The first frame is the Global Reference Frame {A}. A position within this frame at pose n is represented by ${}^{A}X_n \in \mathbb{R}^3_A$, and the origin of this frame is located at the centre of the UR5's base. The second frame is the Tool Point Reference Frame {B}. A position within this frame at pose n is represented by ${}^{B}X_n \in \mathbb{R}^3_B$, and the origin of this frame is located at the centre of the surface of the tool point, as shown in [63]. The pose of the origin of {B} within {A} can also be represented by the vector $(c_1, c_2, c_3, a_1, a_2, a_3)$ where $(c_1, c_2, c_3) \in \mathbb{R}^3_A$ represents a point in space, and $(a_1, a_2, a_3) \in [0, 2\pi]^3$ represents the axis angles of {B} with respect to {A}. The final reference frame is the Checkerboard Reference Frame {C}. A position within this frame at pose n is represented by ${}^{C}X_n \in \mathbb{R}^3_C$, and its origin is located at the bottom left corner of the checkerboard.

While the tool point pose is defined with respect to {A}, the optical centre of the Lytro is defined with respect to {C}. This is because the optical centre's position is estimated using the Matlab Light Field Toolbox v0.4 [64]. The position of {B} within {A} is known, as it is provided by the software to the UR5. To obtain the pose of the Lytro's optical centre within a fixed frame {A}, its position must be calculated with respect to {B} and then transformed to {A}. The distance from the Lytro's optical centre to the origin of {B} when the camera is at the nth pose

defined as $d_n \in \mathbb{R}_A$. Calculating d_n for a number of poses allows the average distance from the Lytro's optical centre, defined as $d_0 \in \mathbb{R}_A$, to be determined. Using the steps outlined below, d_0 can be calculated and used to obtain the position of the Lytro within {A} at each pose.

To calculate d_0, the position of the Lytro's optical centre is first transformed from {C} to {A}. This is possible as the location of {C} can be directly measured within {A}. This measurement resulted in the vector (-1.66, -0.157, 0.085) m. This vector, along with the relationship between frames shown in Figure 3.3 on page 25 can be used to define the homogeneous transformation matrix ${}^{A}_{A}M_C \in SE(3)$, which will transform a point within {C} to a point within {A} using the matrix M also defined in {A}. Here the notation $SE(3)$ refers to the special Euclidean group representing rigid transformations in $\mathbb{R}^3$ [44].

$$ {}^{A}_{A}M_C = \begin{bmatrix} 0 & 0 & -1 & -1.66 \\ 1 & 0 & 0 & -0.157 \\ 0 & -1 & 0 & 0.085 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.1} $$

To calculate the average distance d_0, the camera is moved through $n = 1 \rightarrow 50$ known poses and an image of a checkerboard is captured at each pose. These 50 poses are illustrated in Figure 3.4.

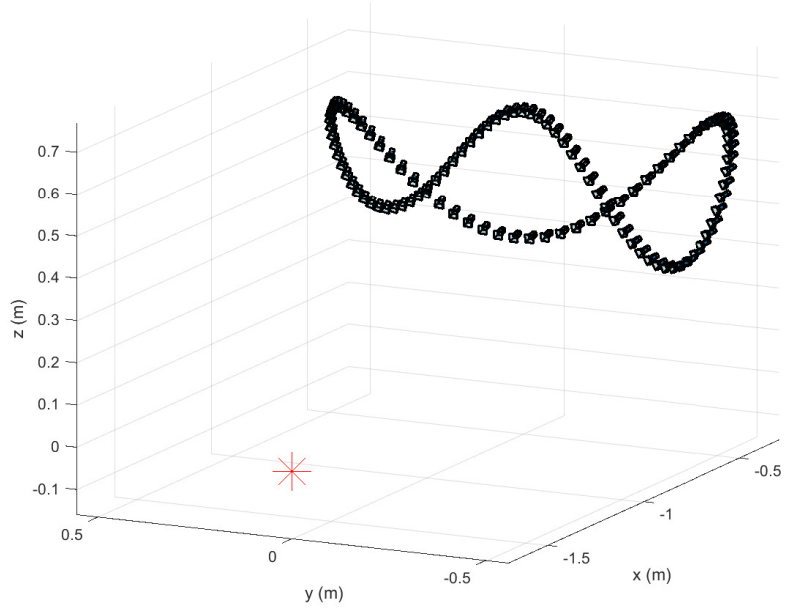


Figure 3.4: Trajectory of UR5 Tool Point over 50 images. The trajectory is adjusted so that the Z_1-axis remains aligned with a focal point, illustrated by * [4].

The position of the Lytro's optical centre within {C} is estimated using the Matlab Light Field Toolbox v0.4 [62]. This software performs a position estimate using a checkerboard-based calibration. Capturing 50 photos of a checkerboard allows d_0 to be calculated for $n = 1 \rightarrow 50$. The position of the Lytro's optical centre is transformed to a position in {A}, and d_n calculated using

$$d_n = \left|\left|{}^{A}T_n - {}^{A}_{A}M_C {}^{C}P_n\right|\right|, \tag{3.2}$$

where n is the number of images used in the calculation, ${}^{A}T_n \in \mathbb{R}^3_A$ is the vector representing the position of {B} at pose n, and ${}^{C}P_n \in \mathbb{R}^3_C$ is the vector representing the position of the Lytro's optical centre at pose n.

Using d_n, d_0 can be calculated via

$$d_0 = \frac{1}{n}\sum_{k=1}^{n} d_n. \tag{3.3}$$

Using the axis angles $(a_{1,n}, a_{2,n}, a_{3,n})$ as defined above, the rotation matrix defined in {A} for pose n used to transform the orientation of the tool point (the origin of {B}) from {B} to {A} $({}^{A}_{A}R_{B,n})$ can be defined as

$${}^{A}_{A}R_{B,n}(a_{1,n}, a_{2,n}, a_{3,n}) = R_x(a_{1,n})R_y(a_{2,n})R_z(a_{3,n}) \tag{3.4}$$

where R_x, R_y, R_z represent the standard 3D rotation matrices around X, Y and Z respectively, and the notation $R_x(a_{1,n})$ indicates that R_x is a function of $a_{1,n}$.

Using the assumption stated above that the distance d_0 should lie exactly on the Z_1 axis, a vector representing d_0, named $x_{z,n}$, can be defined as

$$x_{z,n} = {}^{A}_{A}R_{B,n}[0, 0, d_0]^T. \tag{3.5}$$

The vector $x_{z,n}$ can now be added to the known position of the origin of {B} within {A} to define the position of the Lytro's optical centre within {A} for n.

The position of the optical centre at pose n ($x_{e,n} \in \mathbb{R}^3_A$) can be found via

$$x_{e,n} = {}^{A}T_n + x_{z,n}. \tag{3.6}$$

$x_{e,n}$ is displayed for $n = 1 \rightarrow 50$ in Figure 3.5 on page 29.

The steps used to calculated the vector $x_{e,n}$ are summarised using the Matlab pseudo-code presented below as Algorithm 1. In this pseudo-code $\{[x_{1,n}; y_{1,n}; z_{1,n}; a_{1,n}; a_{1,n}; a_{1,n}]\}$

refers to the vector defining the pose of the UR5 tool point in {A} for image n ($\{[x_{1,n}; y_{1,n}; z_{1,n}\}$ (this corresponds to ${}^{A}TX_n \in \mathbb{R}^3_A$ in Equation (3.2)). Finally, $\{[x_{2,n}; y_{2,n}; z_{2,n}]\}$ refers to the vector defining the position of the Lytro's optical centre in {C} for image n (this corresponds to ${}^{C}X_n \in \mathbb{R}^3_C$ in Equation (3.2)).

Algorithm 1 Average Distance Calculation

procedure DISTANCE CALCULATION
  $\{[x_{1,n}; y_{1,n}; z_{1,n}; a_{1,n}; a_{1,n}; a_{1,n}]\} \leftarrow FILEINPUT$ ▷ import tool point poses
  $\{[x_{2,n}; y_{2,n}; z_{2,n}]\} \leftarrow FILEINPUT$ ▷ import Lytro positions
  ${}^{A}_{A}M_C = [0, 0, -1 - 0.66; 1, 0, 0 - 0.157; 0, -1, 0, 0.085; 0, 0, 0, 1]$ ▷ define rotation matrix
end procedure
for `i=1:n` **do**
  $d_n = \texttt{dnorm}(\{[x_n; y_n; z_n]\} - {}^{A}_{A}M_C\{[\mathrm{x}_{1,n}; y_{1,n}; z_{1,n}]\})$ ▷ array of distances
end for
$d_0 = \texttt{sum}(d)/\texttt{length}(d)$ ▷ average distance
procedure LYTRO POSITION CALCULATION
  for `i=1:n` **do**
    ${}^{A}_{A}R_{B,n} = \texttt{rot}([a_{1,n}, a_{2,n}, a_{3,n}])$ ▷ obtain rotation matrix of tool point
    $x_{z,n} = {}^{A}_{A}R_{B,n}\{[0; 0; d_0]\}$ ▷ average distance vector
    $x_{e,n} = \{[x_{1,n}; y_{1,n}; z_{1,n}]\} + x_{z,n}$
  end for
end procedure

Using this method the average distance between the origin of {B} and the Lytro's optical centre was found to be $d_0 = 10.63$ cm. This correlates well with the Lytro's length of ∼11cm. The vector $x_{e,n}$ defines the location of the Lytro's optical centre within {A} (a fixed reference frame) for each pose n, as required.

With the position of the Lytro's optical centre determined within a fixed reference frame, the UR5 can be given a new set of movements and a dynamic data set can be captured. Each camera pose n will once again have an associated rotation matrix defined by ${}^{A}_{A}R_{B,n}$, which can be used to determine the pose of the camera within {A} ($x_{e,n}$) using the `LYTRO POSITION CALCULATION` procedure described in Algorithm 1. Using this approach a 200 image data set has been captured. Four images within this data set are displayed in Figure 3.6 om page 29.

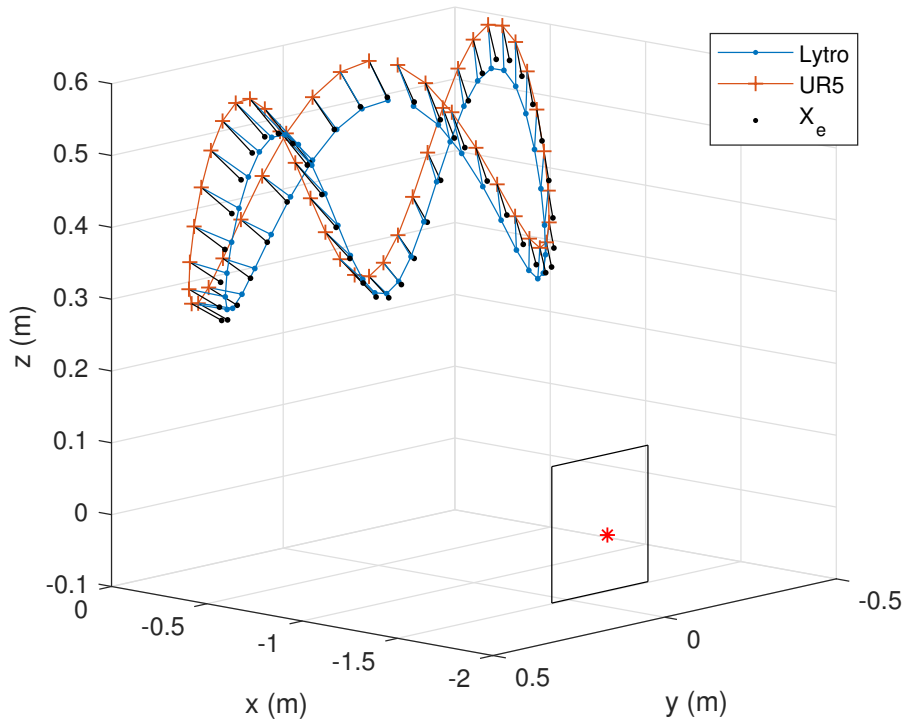


Figure 3.5: The location of the tool point 'UR5' (defined by ${}^{A}T_{n}$) and the Lytro's optical centre 'Lytro' (defined by ${}^{A}_{A}M_{C}{}^{C}P_{n}$). The focal centre that the tool point rotates around is also shown as *, and the location of the checkerboard is defined by the black box. x_e is also plotted for $n = 1 \rightarrow 50$ to illustrate the result of the position calculation.

3.4 Future Data Sets

While dynamic light field data sets can be captured using the software and approach described above, there are some outstanding issues that require discussion.

The distance calculation required to determine the Lytro's position uses the position estimation provided by the calibration regime of the Matlab Light Field Toolbox v0.4 [62]. This toolbox estimates the position over four optimisation iterations, each

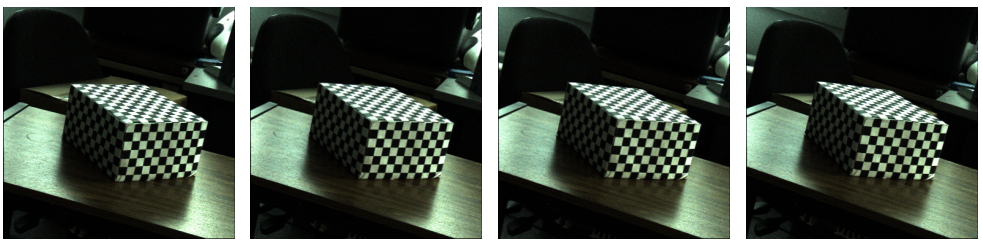


Figure 3.6: Four frames from the 200 image dynamic light field data set.

intended to correct the position estimate based on features such as lens distortion [62]. However, the results of the position estimation appear to diverge from the true position after the first optimisation iteration. Therefore, the position produced by the first iteration is used in the distance calculation. As the first iteration is only an initial estimation, it is likely that there remains some discrepancy between the position estimate and the true position. It is expected that this will be reduced if the subsequent iterations can be implemented correctly.

Efforts to improve the optimisation process included decreasing the number of pixels per micro-lens that were used in the calibration process by increasing the size of the pixel border beneath each micro-lens. It was expected that pixels closer to the edges of the micro-lens would be more susceptible to vignetting, and thus produce less reliable results. Increasing this border appeared to have no effect until it was increased to a radial depth of 5 pixels per micro-lens. A border this large produces results that are much closer to the initial estimate. However, with so few pixels still being used in the calibration it is likely that this change simply reduces the influence of later calibration iterations, rather than improving them. Potential fixes that can be attempted in the future include disabling the bias terms in the distortion model or disabling the third radial distortion coefficient [65]. These terms are part of the Matlab Light Field Toolbox v0.4 [62]. This position estimation issue may also be due to a Lytro firmware update. As the Matlab Light Field Toolbox v0.4 extracts information from the Lytro, a firmware update may have altered the location or structure of this information in such a way that it is now read incorrectly by the toolbox.

The 200 photo data set was captured with the optical axis of the camera always intersecting the same point (the focal point shown in Figure 2.7). However, capturing a data set in this way produces an ill-conditioned calibration problem [66]. This was not realised until after the data set was captured, however further position estimation attempts undertaken with a different setup indicate that this is not the cause of the position estimation problem described above.

If this software is to be used to capture future data sets, the UR5 trajectory should first be updated so that the tool point no longer rotates around a single point. This can be done by editing the `generateUR5Trajectory.m` file that is used by the Lytro Remote Shutter program.

Chapter 4

Physical Data Interception

The following chapter outlines the results of attempts to extract video data from the Lytro via physical interception. The location of the image data path within the Lytro indicates that the most promising extraction location is between the image sensor and image processor. The reasons for this decision are outlined in Section 4.1. The results of performing continuity testing on the ribbon cable connecting the image sensor and processor are presented in Section 4.2. These results include the identification of the physical locations and types of signals passing between these components. The identified signals show that the interface being utilised by the image sensor is a Parallel Pixel Data (PPD) interface. A comprehensive overview of the PPD interface is also provided in Section 4.3. Using a Riser PCB (Section 4.4) and form tool (Section 4.5), a signal analysis (Section 4.6) is undertaken on the PPD interface. This confirms the expected signal locations and types. The results of these sections indicate that it may be possible to physically extract light field video from the Lytro. A list of requirements for an 'Extraction System' that might perform this task is presented in Section 4.7.1, while a conceptual design is outlined in Section 4.7.2.

4.1 Extraction Location

There are two main options available for the physical extraction of the light field data: extracting the data directly from the CMOS sensor, or extracting it en route from the sensor to the processor. A third option that was initially considered involved replacing the CMOS with one that could be easily interfaced with. However, the micro-lens array is attached directly to the CMOS in the Lytro, making this approach unfeasible. These two options are each discussed briefly in this section. It is concluded that the second option; interception during transfer, is the more promising approach.

4.1.1 CMOS Sensor Extraction

Examining the interface of the MT9F002 CMOS sensor [5] indicates that all the connections required to transfer image data to the processor are being utilised for

that task. There are therefore no floating pins that might have the potential to transmit data. Furthermore, the physical location of the CMOS within the Lytro (see Figure 4.1) means that it would be exceedingly difficult to interface directly with it. For these reasons it is concluded that it will be too difficult to extract image data directly from the CMOS sensor.

4.1.2 Data Interception

Intercepting the data before it reaches the processor requires knowledge of the physical location of the relevant wires. Inspecting the dissembled Lytro (dissembled via the steps outlined in Appendix A) indicates that the data must be travelling from the sensor to the processor via a ribbon cable. This cable is displayed alongside the two sections that it connects to in Figure 4.1.

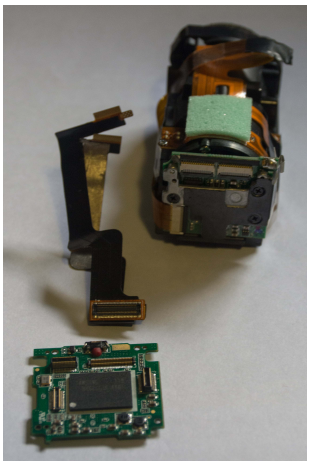


Figure 4.1: Dissembled Lytro showing main board, ribbon cable, and bottom of sensor board attached to lens piece.

Due to this accessible configuration it appears that the best location to attempt to extract data is at the end of the ribbon cable closest to the sensor board in Figure 4.1). This physical interception has been undertaken using the 'Riser Printed Circuit Board (Riser PCB)' discussed in Section 4.4.

4.2 Continuity Testing

Continuity testing involves using a multimeter to check if current can flow between two points. This testing was undertaken between the Lytro's image sensor and main board so as to locate the signals being used to transfer light field image data. The results of this testing include the identification of the most likely type of interface

being utilised by the CMOS sensor, (a Parallel Pixel Data interface), and the physical location of relevant signals. The results are summarised in Figure 4.2 on page 34, and in Figure 4.3 on page 35. The information presented in this section is critical to the success of the final System presented by this thesis, as it will need to physically intercept and process these signals to extract the video data.

The testing indicated that there are 15 CMOS sensor pins electrically connected to the Lytro's main board via the ribbon cable pictured in Figure 4.1 on page 32. The physical location of these signals on the CMOS is illustrated by the black-shaded pins in Figure 4.3 on page 35. The testing also clearly indicated which of these signals are tied to electrical ground, and this information is also included in Figure 4.3.

The locations of the signals found by this testing indicate that the interface being utilised by the CMOS sensor is most likely the Parallel Pixel Data (PPD) interface. This conclusion has been reached by comparing the location of the signals found by the testing to the location of the signals required for the PPD interface, as illustrated in the datasheet of a similar CMOS [6]. This diagram (included as Figure D.1 in Appendix D) very closely matches the MT9F002 CMOS. This can be seen by comparing Figure D.1 in Appendix D to Figure 4.3 on page 35. The datasheet of a similar CMOS is required as the MT9F002 CMOS datasheet does not contain a diagram of its PPD interface pin-out [5]. Appendix D also contains Figure D.2, which describes all the signals that can be expected for the PPD interface. Using Figure D.1 and Figure D.2, the signals found by the testing can be matched to their function. This is shown in Figure 4.2 on page 34.

While the location of the signals found by the testing provides enough information to indicate the most likely interface, it is also clear that not all the signals required for this interface were found. Figure D.1 indicates that signals not found by the testing are the image data signals (`D_OUT`).

Observing the sensor board shows that the signals at these locations on the CMOS pass through 75Ω resistors en-route to the ribbon cable. Performing a continuity test between the resistors and the ribbon cable indicates that these resistors are electrically connected in series to the CMOS and ribbon cable. These resistors are likely included to provide some form of signal conditioning on the image data as it passes from the CMOS to the processor. It is concluded that these resistors are responsible for the data lines not being located by the continuity test. The location of the data lines on the socket of the ribbon cable are included in Figure 4.2.

Comparing the results of the continuity testing to the interface shown in Figure D.1 indicates that the signals `SADDR`, `TEST` and `PIXCLK` are connected to ground. While `SADDR` and `TEST` are not required for the interface to function, `PIXCLK` is required. It is therefore expected that the damaged sensor board (damaged by the inclusion of the form tool discussed in Section 4.5) has a broken clock divider circuit for the `PIX_CLK`. In this case, `PIX_CLK` could appear tied to ground. As `PIX_CLK` is expected to be present in a functioning Lytro, it is included in Figure 4.3 on page 35. `PIX_CLK` is not shown in Figure 4.2 as its location is not known.

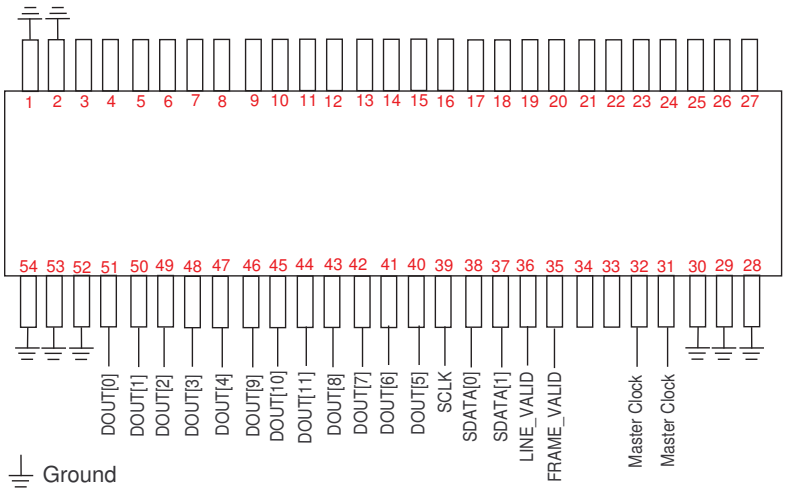


Figure 4.2: The location of the signals found by continuity testing on the socket of the ribbon cable. The socket has the same orientation as shown in Figure 4.6a.

Finally, Figure D.1 indicates where the power pins can be expected to be found on the CMOS sensor. These expected locations can be confirmed by examining the CMOS sensor and the sensor board, as all the power pins are connected to capacitors and use larger traces (equivalent to wires in a conventional circuit) than other connections. This information is included in Figure 4.3 on page 35. The location of these pins exactly match Figure D.1, further validating the relationship between this interface and Lytro's CMOS sensor.

Figure 4.3 on page 35 shows the location of all the signals required for the PPD interface. The CMOS pins found to be connected to the main board via the continuity testing are shaded black, power pins are shaded red, and data pins (although also part of the PPD interface) are shaded blue. While CMOS pin 41 was found to be electrically connected to the main board its function is unknown as this signal does not match any shown in Figure D.1.

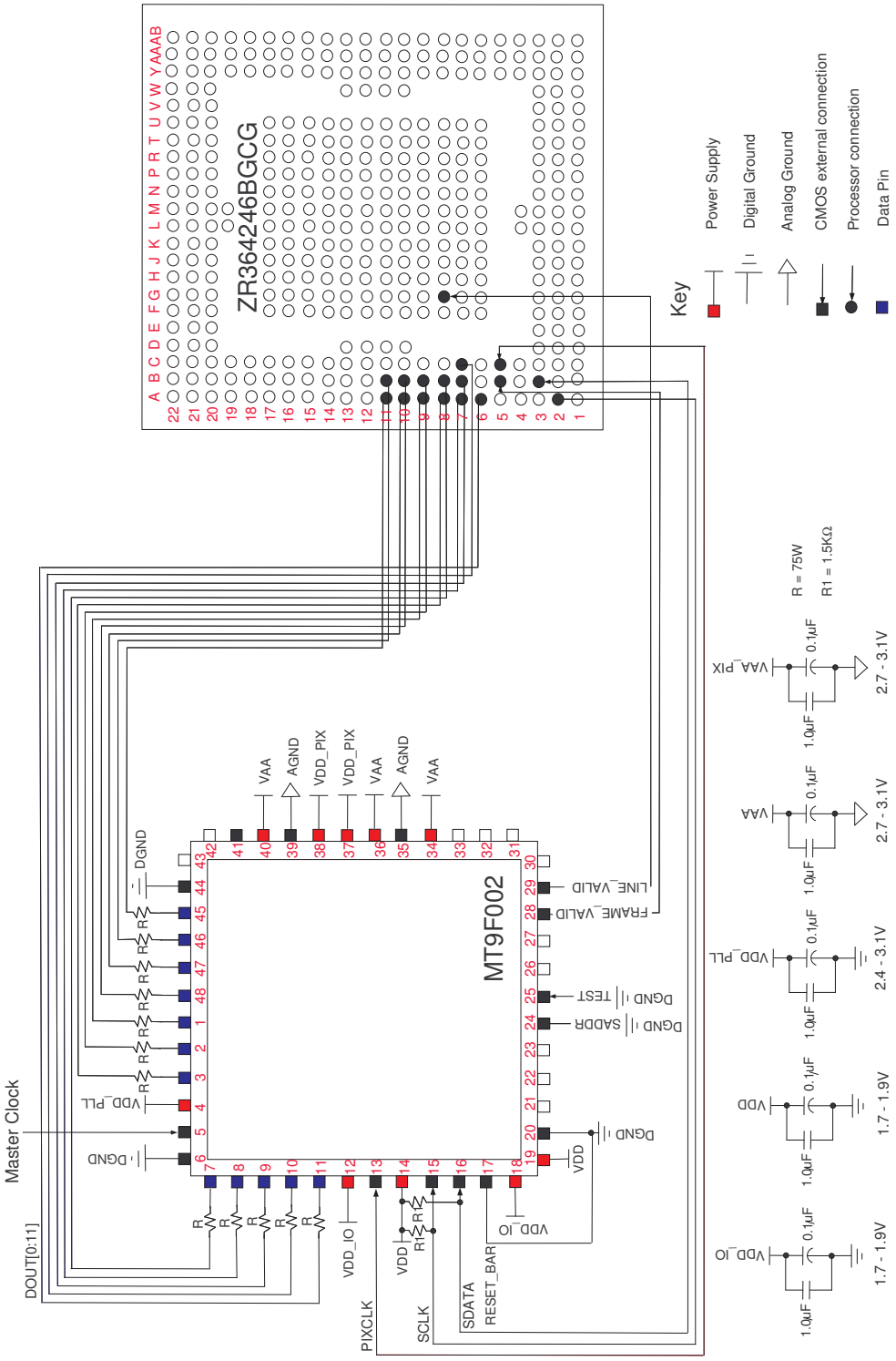


Figure 4.3: Schematic of MT9F002 CMOS sensor & Zoran COACH processor electrical interface (figure described on page 34) [5, 6, 7].

The ribbon cable (shown in Figure 2.7 on page 16) passes all the signals that interface between the CMOS and processor, along with power and ground signals. As these signals pass through the socket of the ribbon cable they also pass through the Riser PCB presented in Section 4.4, and will pass through the Interceptor PCB presented in Section 4.7.2.

This section provides an overview of the signal types and locations of the CMOS's PPD interface. Section 4.3 outlines how these signals are encoded and sent. The signals themselves have also been analysed using the Riser PCB presented in Section 4.4, and the form tool presented in Section 4.5. The results of this analysis are outlined in Section 4.6.

4.3 Parallel Pixel Data Interface

The physical layout of the PPD interface is provided in Section 4.2. The corresponding signal timing and encoding must be known if light field data is to be successfully extracted and interpreted. This section presents an overview of the protocols used by the PPD interface within the Lytro. The protocol of this interface has two main components; the data timing and the data encoding. The data timing refers to the the process of sending and receiving bits, while the data encoding refers to how these bits are interpreted.

The CMOS output data timing is illustrated via Figure 4.4. The output data is synchronised to the 96Mhz `PIX_CLK`, with one pixel (P_n) output per clock period [5]. The data is only output if `LINE_VALID` is HIGH. `LINE_VALID` indicates that a valid row of pixel data is being transmitted. `LINE_VALID` is only HIGH if `FRAME_VALID` is HIGH, although this is not shown in Figure 4.4. `FRAME_VALID` indicates that a valid frame of data is being transmitted.

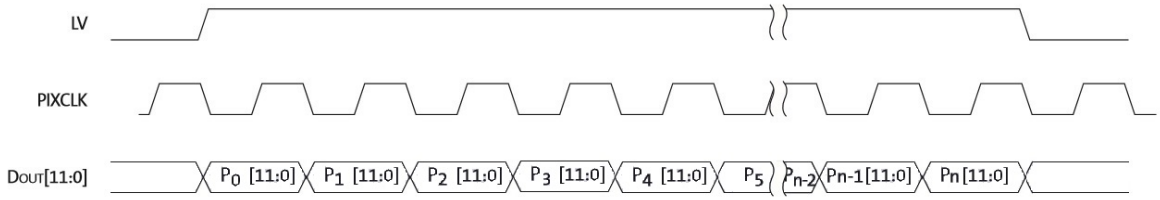


Figure 4.4: CMOS sensor output data timing (signals described above figure) [5].

The input data is sampled synchronously with `PIX_CLK` by the COACH processor. The input data timing of the processor is illustrated in Figure 4.5. Figure 4.5,

displays the names of the input signals according to its data sheet [7]. The signals SEN_CLK, SEN_D[x:0], SEN_VIS, and SEN_HIS correspond to the CMOS signals PIX_CLK, DOUT[0:11], FRAME_VALID, and LINE_VALID respectively. FRAME_VALID outputs HIGH for the duration that a full image is being transmitted to the processor (i.e. it will remain high until all the rows of pixels qualified by LINE_VALID that are required for a full image have been transmitted). GCLKx2 is the global clock.

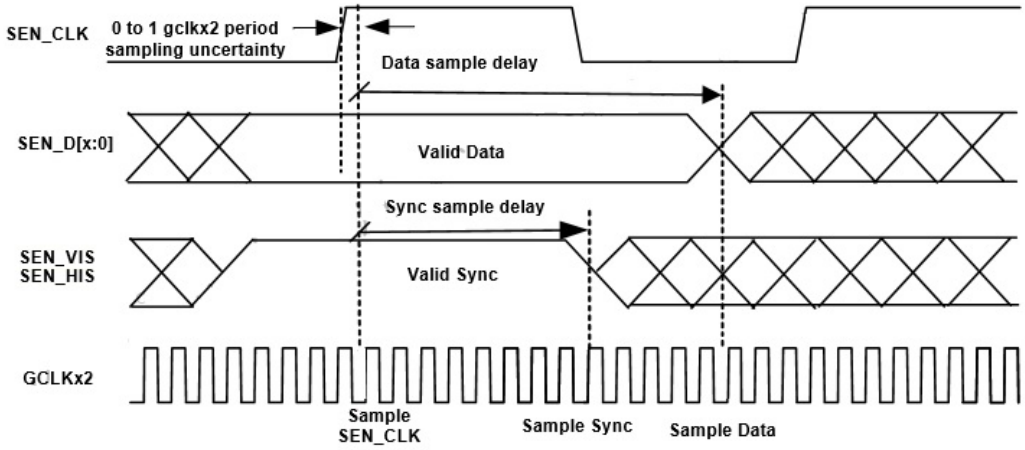


Figure 4.5: COACH processor data sampling (signals described above figure) [7].

Figure 4.4 on page 36 and Figure 4.5 illustrate how information is sent and received between the sensor and the processor. However, these figures do not convey any indication of how the information is encoded within DOUT[0:11]/SEN_D[x:0].

As stated above, the Lytro utilises 12 bits for each pixel on the CMOS sensor. These 12 bits are used to represent the intensity recorded by the sensor at each pixel. Thus, a total of $2^{12} = 4096$ different colours can be represented for each pixel. The intensity values correspond to the primary colours via a Bayer filter with the same configuration shown in Figure 2.1 (i.e. even rows contain green and red pixels, and odd rows contain green and blue pixels). These bits are packaged into the RAW12 data format (RAW image format with 12 bit depth) and stored in a First In, First Out register [5].

Using the above information, the protocol used to transfer the image data to the processor can be summarised as follows: The 12 bits corresponding to one pixel on the CMOS are transferred from the sensor to the processor beginning on a HIGH→LOW transition of PIX_CLK as long as the conditions that FRAME_VALID = HIGH and LINE_VALID = HIGH are met. The transfer will take one period of PIX_CLK to complete, and will continue as a 'bulk transfer' for as long as FRAME_VALID and LINE_VALID remain HIGH [5].

4.3.1 CMOS Sensor Sub-Sampling

Given that the Lytro's CMOS sensor is cropped to 3280^2 pixels [11], and the sensor transfers data at a frequency of 96Mp/s (the frequency of the `PIX_CLK`), it can be determined that the maximum output data frequency (f) at full resolution would be:

$$f = \frac{1}{3280^2/96 \cdot 10^6} = 8.92\text{fps}. \tag{4.1}$$

As the LCD screen is refreshing at 60fps it is likely that the CMOS is sub-sampling while in rolling-shutter mode. The alternative possibility is that the CMOS is operating at a lower frequency, and extra frames are being added between existing frames by the processor. However, this appears less likely as the required frame rate can easily be met via sub-sampling. The MT9F002 CMOS sensor supports sub-sampling of every 1,3 or 7 pixels along its x-axis and/or every 1,3,5,7,15 or 31 pixels along its y-axis [5]. The 60fps frame rate could be achieved by sampling every third pixel on both the x and y CMOS axis, although this is by no means the only possible combination. Such sub-sampling will add extra complexity if this data is successfully extracted by the Extraction System presented in Section 4.7, as it may be unclear how this sub-sampling occurred and thus how to interpret the image data.

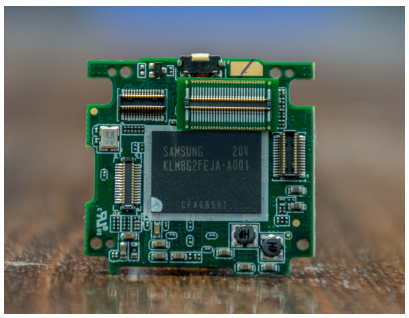
A further concern is that sub-sampling may be occurring in such a way that no light field data is available, for example if only one pixel under each micro-lens were sampled. Given that there are $\sim 360^2$ micro-lenses, this could occur if the sampling in each direction were higher than ~ 9. However, this level of sub-sampling is only available along the CMOS sensor's y-axis. Furthermore, it is possible to re-focus images in real-time by touching the Lytro's LCD screen. While this re-focusing could occur as a result of the CMOS sampling a new set of pixels when the focal plane requires changing, it appears more likely that some amount of light field data is being transferred to the LCD screen. Even if no light field data is available at the extraction location (see Section 4.1 for this location), it may be possible to change the setting of the CMOS if the Extraction System is successfully implemented. This would be possible if the signals being sent from the processor to the CMOS sensor could be replicated by the Field-Programmable Gate Array (FPGA) within the System.

Given these factors, it appears that if data can be extracted by the Extraction System it is likely to contain light field information. Thus, the best approach to confirm this is to attempt to manufacture and test the System.

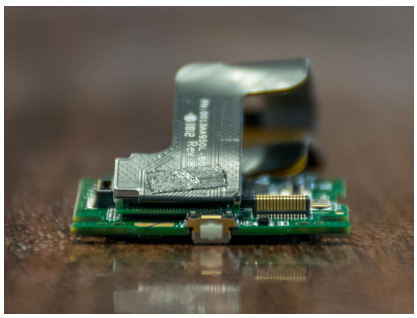
4.4 Riser PCB

'Riser PCB' is the name given to PCBs that are designed to be inserted between two existing components in a circuit. Inserting a Riser PCB between the ribbon cable and main board of the Lytro (see Figure 4.1 on page 32) indicates that such a PCB can be added to the Lytro without breaking it. It also allows a signal analysis to be performed on the ribbon cable when paired with the form tool presented in Section 4.5. Due to the success of the Riser PCB, its design has been incorporated into the design of the Interceptor PCB presented in Section 4.7.2.

The Riser PCB comprises of the head and socket of a Panasonic 54-pin F4S Mezzanine connector placed on either side of a 2-layer PCB [67]. These two components are connected through the PCB using exposed vias (through-holes between PCB layers). With the PCB inserted in the Lytro, its vias could be probed with the assistance of the form tool discussed in Section 4.5. The results of this investigation are discussed with Section 4.6. The PCB is displayed in Figures 4.6a and 4.6b. The PCB design is provided in more detail within Appendix E.



(a) Riser PCB attached to the Lytro's main board.



(b) Riser PCB sandwiched between the main board and ribbon cable.

Inserting the PCB into the Lytro appeared to have no impact on the performance of the camera. The PCB was designed with short traces, thus minimising the impedance added to the Lytro.

The PCB needed to be small with short traces so as to reduce its impedance and allow it to fit inside the camera. Its small size has meant that it could not be accurately probed with an oscilloscope or multimeter. The pins on the socket on the Riser PCB have therefore been 'broken out' on to larger PCB boards using the form tool discussed below in Section 4.5. Combining the form tool with the Riser PCB allowed for a signal analysis presented in Section 4.6 to be undertaken.

4.5 Form Tool

Creating a form tool to break out the pins on the Riser PCB discussed in Section 4.4 has allowed the signal analysis presented in Section 4.6 to be performed. The steps used to manufacture the form tool are presented in Appendix F. Illustrations of this process are also included as Figures F.1a-F.1d.

With the form tool attached to the Lytro the camera still functioned, but the LCD screen displayed only white. This indicates that the addition of the form tool increased the impedance within the Lytro to the point that the image data could no longer be properly transmit to the screen. Furthermore, after some initial signal analysis was performed the Lytro ceased working completely. Given that the `PIX_CLK` signal on this Lytro's sensor board appeared to be tied to ground during the testing undertaken in Section 4.2, it appears that the Lytro failed when the clock divider circuit driving the `PIX_CLK` broke. Such a failure is not unexpected, given the large amount of extra impedance that the form tool added to the Lytro relative to the amount added by the Riser PCB. A second possibility is that an electrical short caused by the form tool could have caused the Lytro to break, and that the `PIX_CLK` issue is unrelated.

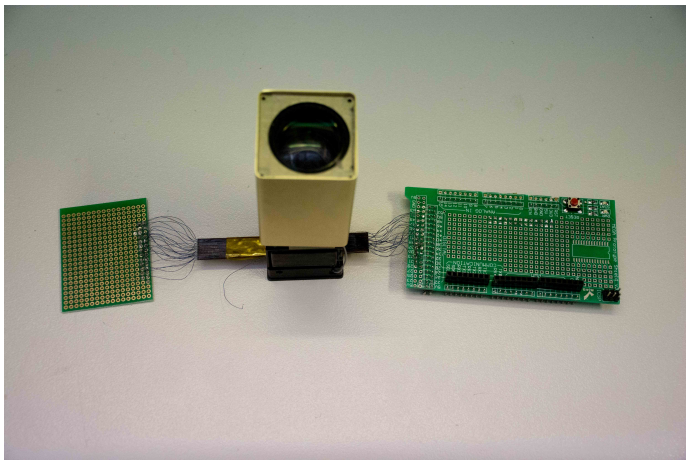


Figure 4.7: Form tool connected to the Riser PCB within the Lytro.

Due to the very small-scale soldering required for this Tool, not all of the wires were properly connected to the Riser PCB. This impacted the signal analysis that could be performed, as it could not be assumed that wires showing no signal were actually floating. These issues led to the signal analysis being less comprehensive than hoped. However, the use of the Riser PCB and form tool still verified that the signals could be intercepted and analysed with a PCB while also still producing the results presented in Section 4.6.

4.6 Signal Analysis

As discussed above in Section 4.5, only a very preliminary signal analysis was undertaken before the Lytro stopped working. The results obtained before this occurred include the validation of the the expected location of the data lines on the ribbon cable socket (as outlined in Section 4.2), and an indication of the different power voltages within the Lytro. Table 4.1 summarises the information obtained using the Riser PCB, form tool and an oscilloscope.

Table 4.1: Signals found on each pin of the Riser PCB. Numbers match those shown in Figure 4.2 on page 34.

Pin	**Signal**	**Pin**	**Signal**	**Pin**	**Signal**	**Pin**	**Signal**
1	-	**15**	5V DC	**28**	-	**42**	18MHz AC
2	-	**16**	5V DC	**29**	-	**43**	18MHz AC
3	-	**17**	5V DC	**30**	-	**44**	18MHz AC
4	-	**18**	5.3MHz AC Square Wave	**31**	-	**45**	18MHz AC
5	-	**19**	1.8V DC	**32**	-	**46**	18MHz AC
6	3.4V DC	**20**	1.8V DC	**33**	-	**47**	17MHz AC
7	3.4V DC	**21**	AC Square Wave	**34**	-	**48**	15MHz AC
8	-	**22**	-	**35**	-	**49**	100kHz AC
9	-	**23**	1.8V DC	**36**	-	**50**	3.3MHz AC
10	-	**24**	3.4V DC	**37**	-	**51**	18MHz AC
11	3.4V DC	**25**	2.4V DC	**38**	-	**52**	-
12	-	**26**	-	**39**	-	**53**	-
13	-	**27**	-	**40**	18MHz AC	**54**	-
14	-			**41**	18MHz AC		

The are several key aspects of Table 4.1 that require attention. Firstly, the AC signals in Table 4.1 all have associated frequency values that were determined via the internal trigger of the oscilloscope. However, the data signals are aperiodic and there is therefore no periodic trigger that can be matched to the signal. Such signals instead require an external trigger. As these signals are synchronised to `PIX_CLK`, this clock would need to be this external trigger. Unfortunately, this could not be done due to the broken `PIX_CLK` discussed in Section 4.2, and the need to do so was not realised until after the Lytro ceased working. Therefore, while the pin locations of the AC signals can be relied on, the frequencies of these signals cannot be. It can be seen that pins 40 to 51 all have some form of AC signal. This matches the expected location of the data lines, as shown in Figures 4.2 and 4.3. Therefore this

analysis provides validation that the data signals pass through these pins.

Comparing the DC voltages in Table 4.1 with the ideal values shown in Figure 4.3 on page 35, it can be seen that they do not match. The expected voltages are either 1.7-1.9V or 2.4-3.1V. There are several possible reasons why this discrepancy may have occurred. These possible reasons include that the wrong reference voltage may have been used to measure the voltages, that there may have been a poor connection with the oscilloscope, that the board may already have been damaged in a way that affected the voltages. Further investigation will be necessary to determine the true cause of this discrepancy. Finally, it is unclear what the two square wave signals on pins 18 and 21 might be.

4.7 Extraction System Conceptual Design

The second PCB designed for this project is named the 'Interceptor PCB'. A model of this PCB is displayed in Figure 4.9 on page 48. The role of the PCB is to physically intercept the signals passing along the ribbon cable, and transfer them to a Xilinx Artix 7 XC7A15T Field-Programmable-Gate Array (FPGA). This FPGA is located on a ZTEX USB-FPGA-Module 2.14 board [8]. The USB-FPGA Module includes a USB 3.0 interface that can potentially be used to pass the image data to an external PC via the FPGA. Together, these components form what has been called the 'Extraction System'. An initial list of requirements for the Extraction System is provided in Section 4.7.1. A conceptual design of the Extraction System is presented in Section 4.7.

4.7.1 Requirements Analysis

The initial list of requirements presented has been updated in an iterative process as the conceptual design outlined in Section 4.7.2 has developed. The design choices that have led to additional requirements are outlined in Section 4.7.2. The requirements are presented in Tables 4.2, 4.3, 4.4, 4.5 and 4.6. These tables respectively outline the requirements for the following areas: 1. The interface between the FPGA and the Lytro's CMOS sensor, 2. The physical system, 3. The functionality of the FPGA, 4. The interface between the FPGA and the USB interface, and 5. The overall, general requirements. When applicable, these tables have a 'main' (integer numbered) requirement, while subsequent requirements in the table are included to refine this main requirement. This is not always appropriate and some tables do not include a main requirement. Some of the requirements are included based on choices or preferences, while others are mandatory for image data to be extracted.

Such requirements are marked with 'M'.

Table 4.2: FPGA-PCB interface requirements.

Reference	Description	Comment
1	The FPGA should have firmware that is compatible with an MT9F002 CMOS sensor.	
1.1	The FPGA must be able to receive an input pixel clock of 96MHz.	M.
1.2	The FPGA must have an available input for the `FRAME_VALID` signal.	M.
1.3	The FPGA must have an available input for the `LINE_VALID` signal.	M.
1.4	The FPGA must be capable of receiving transfer of parallel data using the signals `FRAME_VALID` and `LINE_VALID` for flow control and frame synchronisation.	M. See Figures 4.4 and 4.5 on pages 36 and 37.
1.5	The FPGA must be compatible with the RAW12 data format.	M.
1.6	The FPGA must have 12 input pins available to interface with the Parallel Pixel Data Interface of the CMOS.	M.
1.7	The FPGA must be capable of receiving data from a 12 bit parallel interface.	M
1.8	The FPGA must be able to receive 12 bits per period of a 96MHz input clock.	M.
1.9	The FPGA must have an internal clock with a frequency greater than $96 \times 2 = 192$MHz.	M. To ensure accurate sampling of the image data.
1.10	The FPGA must be able to correctly interpret the HIGH and LOW signal levels sent from the CMOS	M.
1.11	The FPGA must be able to represent data in the same way as the CMOS.	M. E.g. twos compliment, etc.
1.12	The FPGA must be compatible with the line code of the CMOS.	M.

Table 4.3: Physical PCB requirements.

2	The PCB must transfer signals from the Lytro to the FPGA.	M.
2.1	The impedance added by the Extraction System must be less than 68.5pF (the output load of the CMOS [5]) plus a currently unknown constant.	M.
2.2	The PCB must be designed such that its connector can be inserted between the ribbon cable and main board.	M.
2.3	The dimensions of the PCB must ensure that all components of within the Lytro can remain connected.	M.
2.4	The PCB must include a power interface for the FPGA.	
2.5	The PCB must have a USB port compatible with USB 3.0 or above.	M. Due to Requirement 4.2.
2.6	The PCB must include a Panasonic Mezzanine 54 pin header and socket [67] to physically interface with the Lytro.	M.
2.7	The data lines on the PCB must all have the same length.	M. To synchronise the parallel data.
2.8	The PCB must be designed in such a way that another PCB containing the FPGA can be connected.	M.

Table 4.4: FPGA requirements.

3	The FPGA must be capable of receiving image data and transferring it to an external device.	M.
3.1	The FPGA must be capable of transforming a 12 bit parallel input data stream into a serial data output stream.	M.
3.2	The FPGA must be capable of transforming the input data into the output format at a bit rate of $96\text{Mp/s} = 96 \times 12 = 1.152\text{Gbit/s}$	For real-time data transfer.

3.3	The FPGA must be capable of transferring data to a USB interface at a rate of 1.152Gbit/s	

Table 4.5: USB interface requirements

4.1	The Extraction System must have a USB interface for streaming light field data to a remote PC.	
4.2	The USB interface must be capable of transferring light field data as a bulk transfer at a bit rate of 1.152Gbit/s.	This limits the USB interface to USB 3.0 and above [68].)

Table 4.6: General requirements.

5.1	The Extraction System must be as cheap as possible.	
5.2	The components of the Extraction System, and its overall design should be chosen such that the time taken to complete the design is minimised.	
5.3	The components of the Extraction System should all have available data sheets that provide PCB layout information and example designs.	
5.4	The components that are chosen for the Extraction System should have strong online communities.	
5.5	The components that are chosen for the Extraction System should not be undergoing a phase-out process.	

4.7.2 Conceptual Design

The requirements listed in Section 4.7.1 address the transmission of the image data and the functions that the Extraction System will need to perform. While the requirements regarding data transmission can be ascertained from Section 4.3, the functional requirements depend on the intended use of the Extraction System. In order to reduce complexity, it has been decided that the System will only be designed to stream the data, rather than perform any processing. This means that full frames of the video data do not need to be stored on the FPGA. This removes the need for specific memory-based requirements. It is preferable (but not mandatory) that

this transfer happens in real-time. USB has chosen as the transfer method, despite some WiFi routers being capable of such rates [69]. As either transfer method could potentially work, USB has been chosen so that a USB-FPGA board can be used. There do not appear to be any similar 'WiFi-FPGA' boards.

An FPGA has been chosen as the processor for the Extraction System. The processor could have been either a soft-core or a hard-core microprocessor, however, it proved difficult to find a hard-core microprocessor that met Requirements 1.6-1.9 and 3.3. For example, neither of the Texas Instrument OMAP [70], the Fujitsu Milbeaut M4 [71] or the Panasonic MN101/MN103 [72] have a USB 3.0 interface. Additionally, it was found that none of the On Semiconductor processors [73] were capable of receiving input clock speeds greater than 30MHz, the Sanyo LC82102/LC82102W is only capable of 6-bit resolution, and while the Samsung Exynos 5250 [74] has a USB 3.0 interface, it does not seem to have a parallel pixel interface. This is by no means an exhaustive overview of the available image processors, and it is highly likely that there are some available that would meet the requirements (the Zoran COACH would work if it were still available). However, it has been decided that rather than attempt to find an appropriate microprocessor, a better approach is to pick a suitable soft-core processor that can be adapted to meet the requirements. The specific type soft-core processor that has been investigated is the Field-Programmable Gate Array (FPGA).

Signal processing is a very common use of FPGAs, which are suitable devices for such tasks due to their highly parallel architecture and reconfigurability [75]. Literature including [76, 77, 78, 79] have found FPGAs to be successful alternatives to Digital Signal Processors (DSPs) and Application-Specific Integrated Circuits (ASICs). FPGAs are even being used in light field video applications already. For example, in 2000 Wilburn *et al.* designed a light field video camera using an image sensor array, that implemented FPGAs as the interface between the image sensors and processor [80]. This is the same application that is proposed in this project. FPGAs can be programmed to process parallel signals with a depth that is limited only by the number of input pins [76]. This means that Requirements 1.2, 1.3, 1.4, 1.6 and 1.7 are satisfied by FPGAs generally.

The FPGAs that have been considered are the three main FPGA families; Altera, Xilinx and Lattice. Due to their soft-core design, typical information such as 'maximum clock speed' is not available for FPGAs. Instead, similar applications had to be examined to determine which FPGAs might be appropriate for this task. Of the previous research that has been done into processing of high-volume data with

FPGAs, a large number of sources appear to use either Xilinx [77, 78, 80, 81, 82] or Altera [83, 84]. The final FPGA was chosen from these two companies using the requirements listed in Table 4.6 and the insight provided by the literature. The chosen FPGA is the Xilinx Artix 7, part number XC7A15T [85].

Xilinx has been chosen due to the literature evidence that suggests that it might be suitable. Only current generation Xilinx FPGAs (generation 7) have considered in an effort to meet Requirements 5.4 and 5.5. Of the four families of generation 7 FPGAs; Spartan, Artix, Kintex, and Virtex, the Artix has been chosen as it is cheap and optimises transceiver speed compared to the other families [86]. Transceiver speed is critical in order to reach the necessary data rates and thus satisfy Requirement 3.3. Furthermore, unlike the other three generation 7 families, an FPGA board already exists for the Artix that contains a USB 3.0 port (see Figure G.1), thus satisfying Requirement 4.2. The USB-FPGA board called the ZTEX USB-FPGA-Module 2.14, also ensures that Requirement 5.2 is met, and makes Requirement 5.3 redundant. This USB-FPGA board has therefore been chosen to be part of the Extraction System. A block diagram, function diagram and pin assignment table for this board are located in Appendix G. Further information can be found at [8].

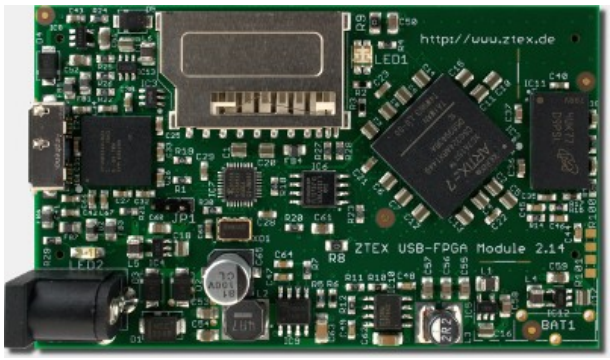


Figure 4.8: ZTEX USB-FPGA-Module 2.14 [8]

Input pins A3→A14, A18→A30, B3→B14, B18→B30 on the FPGA board have been chosen to interface with the Interceptor PCB (see Table G.1 in Appendix G). This provides 50 pins. Four of the ground pins on the ribbon cable socket (see Figure 4.2 on page 34) will therefore not be connected. The trace length from the input pins to the FPGA are listed here [8]. The trace lengths on a new iteration of the Riser PCB will need to be varied so that the overall length matches, thus meeting Requirement 2.7. This can be done using the list of trace lengths available for the USB-FPGA board [87]. The current design of the Interceptor PCB, is shown below in Figure 4.9, with further information provided in Appendix H.

The 2x32 header that will fit into the vias (through-holes) shown in Figure 4.9 is the male header that pairs with the female header on the back of the ZTEX USB-FPGA-Module 2.14 (See Figure G.2). This header is available here [88]. After the traces are aligned correctly this PCB can be manufactured. Due to the small vias (through-holes) on the PCB, very few companies are capable of producing it. One company that can perform this task is PCBCART [89]. Aside from varying the trace lengths, the other key next step is to write software that will allow the FPGA to receive and interpret the data intercepted by the Interceptor PCB, translate this data to a serial data stream and output this stream via the USB interface on the USB-FPGA board.

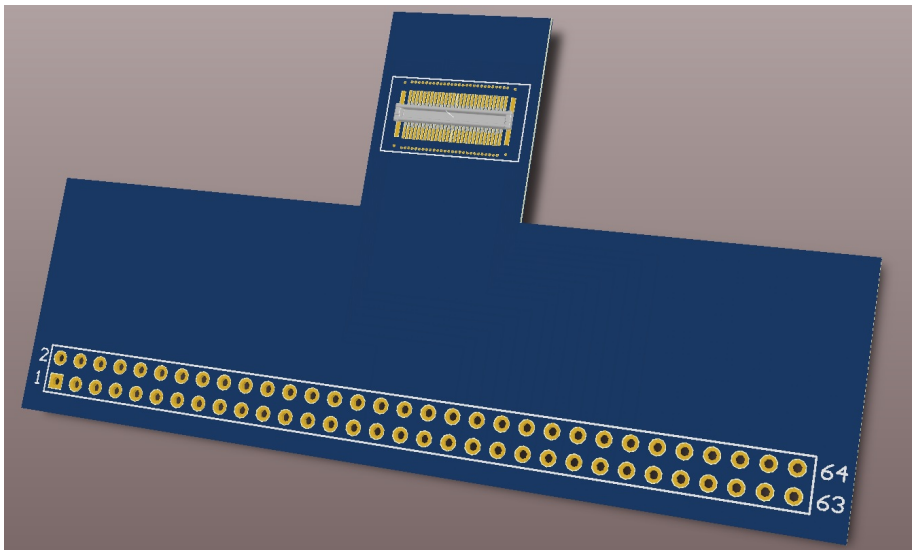


Figure 4.9: 3D view of PCB designed to interface with ZTEX USB-FPGA-Module 2.14 (image created using Altium Designer software package).

If the Extraction System can be made to meet all the requirements presented in Section 4.7.1, it is hoped that it will successfully extract light field video from the Lytro. There are areas of uncertainty in this approach, such as the maximum impedance the System can add to the camera before it ceases working, and the unknown sub-sampling that appears to be occurring at the CMOS sensor. However, it appears that the best way to determine if the System can succeed is to manufacture and test it.

Chapter 5

Conclusions and Further Work

A software package called 'Lytro Remote Shutter' has been developed. This software allows the Lytro to be remotely triggered in a ∼0.54fps loop, and to interface with a UR5. This interface allows the Lytro to be used with a UR5 to capture dynamic light field data sets with associated camera pose information. Using this software, a 200 image dynamic light field data set has been captured.

While the estimation of the camera pose produced by this software appears to be reasonably accurate, the error in this estimation is yet to be quantified. There is also a known issue involving this estimate, whereby the accuracy appears to decrease as optimisation is performed. If an error check on the position estimation indicates that the result is too inaccurate, the diverging estimation may need to be rectified. Potential ways that this may be attempted are outlined at the end of Chapter 3. Furthermore, it was discovered upon capturing the data set that orienting the camera such that it always points toward the subject produces an ill-conditioned calibration problem. This is problematic as the software relies on a calibration regime to perform the position estimation. Despite this connection, further investigation has suggested that this error is not the reason for the divergence discussed above. However, this error will also need to be rectified before further data sets are captured. Suggested steps to perform this are provided at the end of Chapter 3.

The Lytro Remote Shutter software package provides a new tool that allows the Lytro to be used to test and develop applications of light field video. The associated camera pose information allows the software to be applied to dynamic light field video applications. The ∼0.54fps video can also potentially be used for static applications.

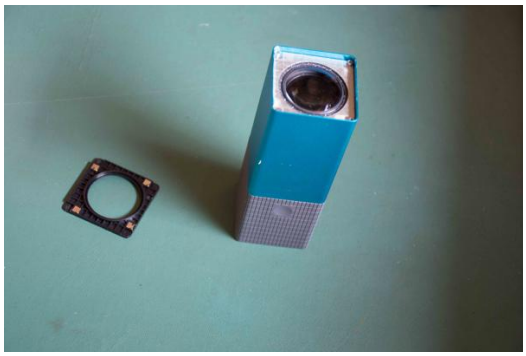
The conceptual design of an Extraction System that has the potential to extract video from the Lytro has also been developed. This System includes an 'Interceptor PCB' and a ZTEX USB-FPGA-Module 2.14 board containing an Xilinx Artix FPGA. A preliminary list of requirements for this System has also been produced.

While the video stream visible on the Lytro's LCD screen was not successfully extracted during this project, the possibility of achieving this in the future has not been ruled out. This thesis indicates that it is possible to physically intercept the light field video data within the Lytro and presents a method of doing this. Key next steps regarding this design include developing software for the FPGA that will allow it to receive the parallel pixel data from the Interceptor PCB, translate it to serial pixel data, and transfer this data via USB to an external device. Before this can be done, the traces on the Interceptor PCB will need to be varied so that all paths from the interception location within the Lytro to the FPGA are the same length. This will ensure that the image data remains synchronised. When the trace lengths are equal, the Interceptor PCB can be manufactured. Once the PCB has been manufactured, the USB-FPGA module bought, and software created for it, it will be possible to test this System.

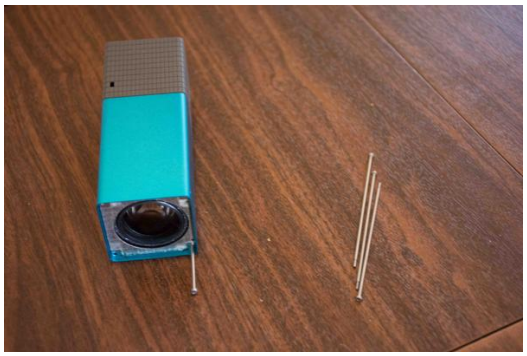
If the Extraction System successfully extracts light field image data, it will then need to be interpreted. This task is likely to prove more difficult due to the sub-sampling discussed in Chapter 4. If the signals extracted by the System can be correctly interpreted, it may be possible to eventually by-pass the Lytro's processor completely. This would lead to a camera system comprising of the Lytro's lens system and image sensor, combined with the Artix FPGA, which would be used to control the image sensor and process/transfer the image data.

This thesis aimed to find new ways to utilise or adapt the Lytro to test and develop applications of light field video. The Lytro Remote Shutter software package represents one such method of doing this. If successful, the Extraction System will provide a second method.

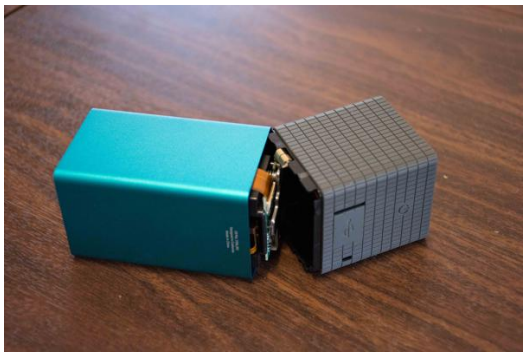
Lytro Disassembly



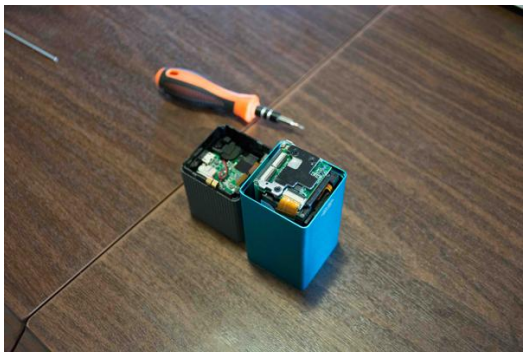
1. Heat was applied to the front of the Lytro, loosening the adhesive tape and allowing the lens protector to be prised off.



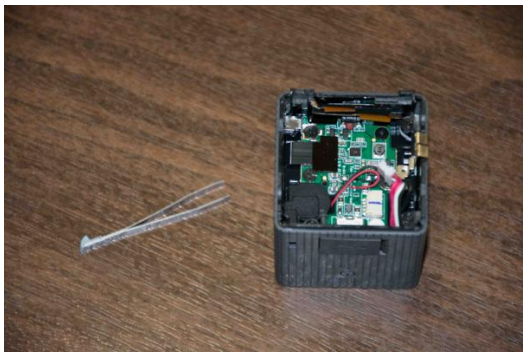
2. The four long screws beneath the lens protector were removed.



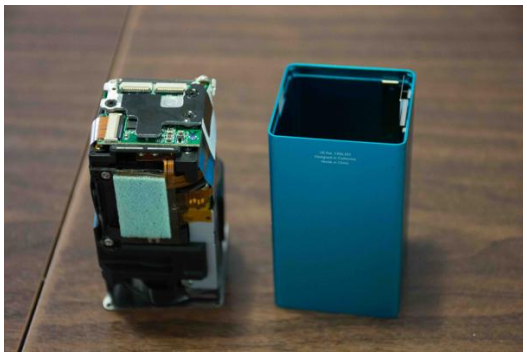
3. The two halves will not split apart, they're still held on by 2 bits of ribbon and a plastic tab. The ribbons were slid out using a flat head screwdriver, allowing the two halves of the Lytro to be separated. Note that the bottom cable is soldered to the sensor board, and needs to be plucked off.



4. With the ribbons remove, the pieces could be slid apart.

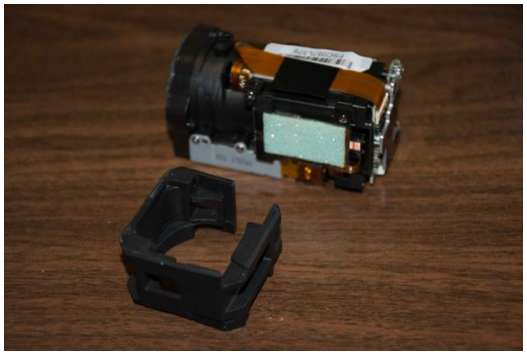


5. The power cable was removed from the LCD half with tweezers.



6. The lens was slid out of its casing.

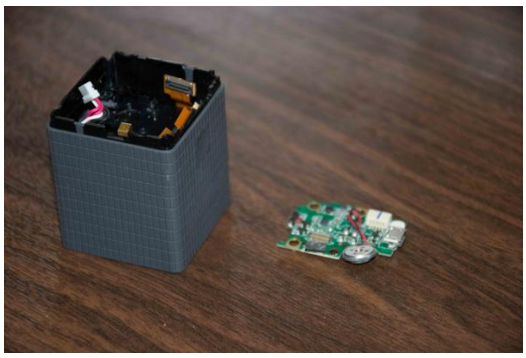
7. A metal piece was removed.



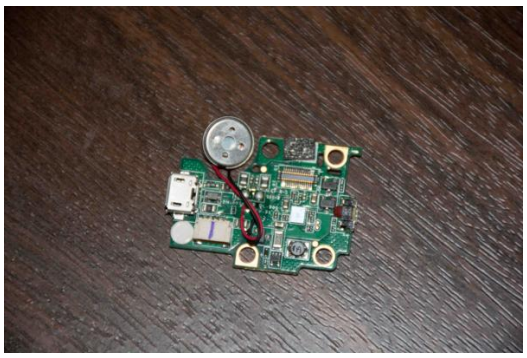
8. A rubber piece was removed.

9. 8 screws and rubber cap were removed from the USB Board.

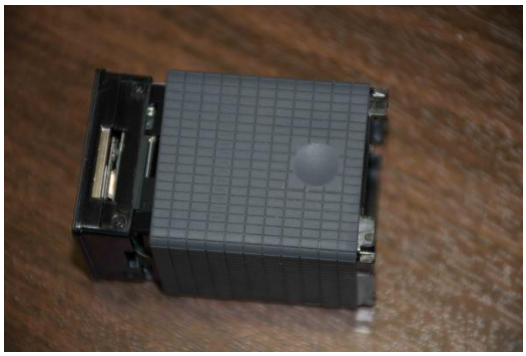
10. A 9th screw was removed.



11. The USB board was removed. It was pulled out after the black cable that is still attached was prized upward. The piezzo buzzer also needed to be prized up as the board was removed.



12. The USB Board.



13. The rest of the screen half of the camera was pushed out of its case. This was the hardest step. Once it came out a little way, a ribbon cable was exposed just behind the screen. This cable is the only thing holding the inside and outside together. Prizing this out should allow the two parts to come apart easily. I broke this cable, although this only affected the zoom function and the camera still worked.

14. The LCD half of the camera removed from it's casing.

15. Unscrewing the four screws from the Main Board allowed it to be removed.

16. The Main Board and LCD Board attached to battery pack.

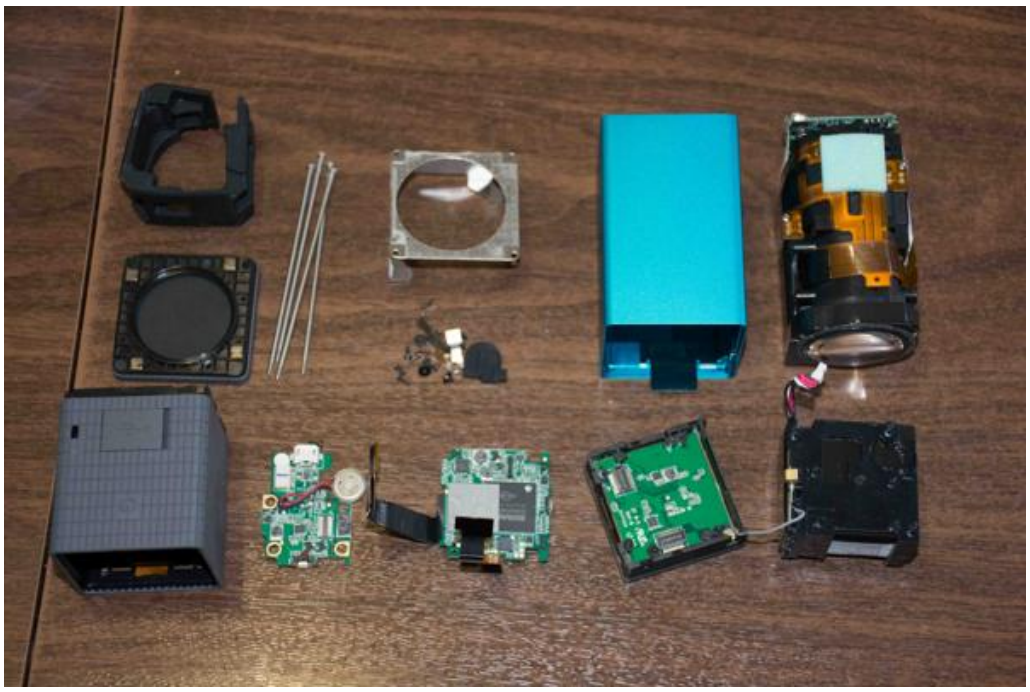


Figure A.9: The dissembled Lytro. Top row: rubber piece that attaches to outside of lens, 4 screws holding the two halves of the Lytro together, glass lens protector, cover of the lens half of the Lytro, the lens piece. Middle row: lens cover, screws, small rubber piece and small metal part attached to a screw. Bottom row: cover of bottom half of Lytro, USB board, main board with ribbon cable, battery pack with LCD screen and board attached.

Communication Examples

USB command example.

Table B.1: USBMS example command. The command is 'return selected file size' command described in [9].

bit → **Byte ↓**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0-3	'USBS'							
4-7	'0'							
8-11	'65536'							
12	'0x80'							
13	'NS'				'0'			
14	'NS'				'12'			
15-30	'0xc6 00 00 00 00 00 00 00 00 00 00 00'							

Where NS = 'not set'.

The protocol outlined in Table 3.1 is described below [9]:

- dCBWSignature: signature identifying packet as CBW, (string "USBC")
- dCBMTag: generated id of the command
- dCBWDataTransferLength: expected length of the data returned by the device
- bmCBWFlags: last bit is direction (0–host to device, 1–device to host in byte)
- bCBWLUN: Logical Unit Number (0 for Lytro)
- bCBWCBLength: length of the command block (12 for Lytro)
- CBCBW: command block - an actual SCSI command

Wifi command example.

Table B.2: Wifi communication example. The command is the 'shutter trigger' command described in [2, 11].

bit → Byte ↓	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0-4	'58530'								'5678'							
4-8	'1'															
8-12	'1'															
12-14	'20'				'0 0 0'			NS	NS	NS	NS	NS	NS	NS	NS	NS
14-18	'62420'								'0xbf2a'							
18-20	'0'															
20-24	'AF 55 AA FA'															
24-28	'0'															
28-32	'1'															
32-34	'C0x00'															
34-48	'00 00 00 00 00 00 00 00 00 00 00 00'															
48-62	'NS'															

The protocol outlined in Table 3.2 is described below [11, 12]:

- Source Port: the sending port (58530 for Lytro)
- Destination Port: the receiving port (5678 for Lytro)
- Sequence Number: flag (1 for Lytro)
- Acknowledgement Number: flag (1 for Lytro)
- Data Offset: the size of the TCP header (20 bytes)
- Reserved: set to 0
- NS, CWR, ECE, URG, ACK, PSH, RST, SYN, FIN: 1 bit flags
- Window Size: the size of the receive window
- Checksum: used for error checking
- Urgent Pointer: flag (0 for Lytro)
- Signature: identifies receiving device (AF 55 AA FA for Lytro)
- Content/Buffer Length: length of content or buffer
- Flags: flag indicating output or input
- Command: device command

- Parameters: device parameters
- Optional Payload: optional payload to be sent or received by device

Appendix C

Lytro Remote Shutter User Manual

C.1 Acknowledgements

This software package utilises software from a range of sources. The remote triggering and downloading of photos from the Lytro light field camera was made possible thanks to a DLL provided by Jan Kučera. More information on this software is available on his website [11] and thesis [2]. Calibrating the light field files and decoding them into a user-friendly format was done using the Matlab Light Field Toolbox v0.4 by Donald Dansereau [64]. Generating a trajectory for the UR5 and calibrating the UR5 with the Lytro was made possible thanks to Matlab code provided by a previous ANU Engineering undergraduate student, Montiel Abello. Finally, communicating remotely with the UR5 was made possible thanks to example Matlab and Polyscope script code by Fereshteh Aalamifar [90] and Long Qian [91].

C.2 Overview

This software has been compiled into a Windows Executable using Visual Studio. It is written in C#, but also includes a range of Matlab functions and a Polyscope program to be run on a UR5 robot arm. The software is capable of several triggering configurations, and also provides downloading and processing functionality. These functions are briefly explained below. The software package is available online at `https://github.com/kennege/Lytro-Remote-Shutter`.

C.2.1 Triggering Options

The software allows for three methods of triggering the Lytro. Option 1 will triggering the camera every 1.5 seconds for a time period that is specified by the user in the `config.txt` file. Option 2 allows the user to trigger the shutter remotely by pressing `SPACEBAR`. Option 3 will trigger the camera while simultaneously moving a UR5 robot arm through a predetermined range of motion. Section C.3.8 outlines how the trajectory implemented by the UR5 can be set up.

C.2.2 Downloading

Once all photos are taken, the photos are downloaded from the Lytro to the user's computer via wi-fi. The location of the downloaded files is explained in Section C.6. There are a range of photo file formats that are available to choose from, and any combination of these can be downloaded by using the appropriate combination of keywords in the `config.txt` file. These file options are detailed in Section C.6. As each photo completes downloading, it is deleted from the camera. This is done to ensure that the correct photos are downloaded if consecutive runs are undertaken.

C.2.3 Processing

Once all the photos have been downloaded to the user's computer, instructions are sent to Matlab's command line to run the Light Field Toolbox mentioned in Section C.1. This process accesses the downloaded files, decodes them, calibrates the Lytro using the white image data discussed further in Section C.3.2, then rectifies the decoded images using this calibration data. This process, along with the location of all the relevant files is outlined in Section C.7.

C.2.4 Results

Upon completion, the user will have access to the original and decoded light field files. All the photos will have been deleted from the camera, allowing for the software to be re-run immediately.

C.3 Before Running This Software

C.3.1 Physical Setup

For full functionality of this software, the user requires a PC, a UR5 robot arm and a first generation Lytro light field camera. The PC and Lytro need to be connected via wi-fi, and the PC and UR5 need to be connected via Ethernet. For the PC and UR5 to communicate, it must be ensured that they are within the same IP subnet mask.

C.3.2 Download Lytro Desktop

Download and install Lytro Desktop from [59]. Lytro Desktop will back up your files and download the white image files that are used by this program to calibrate the light field images. In order for these files to be downloaded, you only need to

plug the Lytro into your PC via USB while Lytro Desktop is installed. No further steps are required.

C.3.3 Download Matlab

Matlab is required in order to decode and calibrate the light field images, and communicate with the UR5. Matlab can be downloaded from here [62]. If you do not have a Matlab licence and wish to skip the UR5 communication, decoding and calibration steps, this can be done using the `config.txt` file discussed in Section C.3.9.

C.3.4 Loading `Lytro.urp` on the UR5

In order for the UR5 to respond to movement commands from this software package, the file `Lytro.urp` must be loaded on to the UR5 Polyscope. This can be done by saving the file to a USB and inserting the USB into the UR5 Polyscope device. The user can then choose the 'Load Program' option, navigate to the file on the USB, and open it. If you receive a warning that the software was written for an earlier version of Polyscope, accept this and continue.

C.3.5 IP Subnet

For communication between the user's PC and the UR5 to be successful, these devices must be within the same IP subnet mask. To check the IP address of the UR5, navigate to 'Setup Robot', 'Network Settings', then choose 'Static Address' and enter the address and subnet.

The addresses used in the initial setup were: PC IP: 192.168.100.2, PC Subnet: 255.255.255.0 PC port: 30000, UR5 IP: 192.168.100.2, UR5 subnet: 255.255.255.0 UR5 port: 30000.

Note that the user will need to update the IP address within the `Lytro.urp` file (see Section C.3.4) if they wish to use a PC IP address other than the above. Furthermore, if the user wishes to use a different port, they will need to change this both within the `config.txt` file, and the `Lytro.urp` file.

C.3.6 Calibrate the Lytro

In order to determine the Lytros intrinsic matrix and allow for the rectification of images [92], the Lytro camera used with this software should be calibrated. This can

be done using the Light Field Toolbox v0.4. This calibration process requires a set of images of a calibration checkerboard. This set should contain at least 10 images in a range of diverse poses, that all contain the whole checkerboard, and maintain the same zoom and focus settings [92]. To perform this calibration, the user needs to undertake the following steps.

1. print off a checkerboard (high density, such as 19x19 squares will yield better results).
2. Measure the length and width of the checkerboard squares.
3. Ensure that Option 1 in the `calib.txt` file is set to '1 or 2', and that Option 2 is set to 'Y'.
4. Ensure that Option 7 is set to 'Y,[x,y],[x1,y2]', where [x,y] refers to the number of corners on the checkerboard (length×width), and [x1,y1] refers to the length and width of the checkerboard squares.
5. Take at least 10 images of a checkerboard with known dimensions. Ensure that the entire checkerboard is visible in all images.

Upon performing the steps outlined above, 10 images will be saved to the directory \<pwd>\LFToolbox0.4\LFToolbox0.3_Samples1\Cameras\<Camera-Serial-Number\01

Lytro Remote Shutter requires access to the Lytro cameras white image files in order to perform the calibration. These are automatically downloaded when the Lytro is connected to a PC via USB. Therefore, before running this calibration routine, the user must first connect their camera to a PC and allow these files to download. Lytro Remote Shutter searches for these files when it runs this routine.

C.3.7 Calibrate Lytro and UR5

The Light Field toolbox and the UR5 pose data give us two different sets of 3D spacial coordinates. We need a set of coordinates that represents the position of the optical centre of the Lytro with respect to some fixed reference point. This would be possible just with the Light field toolbox if we were only using this software to measure planar checkerboards. This is because the toolbox provides us with coordinates relative to this checkerboard, and if we want to measure any other type of object the spacial coordinates calculated by the Toolbox would not be accurate.

To overcome this issue, the Lytro can first be attached to the UR5, and used to measure a planar checkerboard. This set up provided both sets of coordinates: a set corresponding to the position of the UR5 gripper with respect to the UR5 base,

and a set corresponding to the position of the Lytro's optical centre with respect to the checkerboard. By following the process outlined in Section 3.3 of this thesis, these sets of images can be used to determine the offset vector between these two reference frames.

As the external camera parameters that we need are the relative rotation and translation between images, we can do the following. Set the initial gripper position as the reference point. Calculate the rotation and translation of each subsequent image relative to this reference point.

C.3.8 UR5 Trajectory

Note: If you do not wish to change the trajectory of the UR5 from the default (see Figure C.1), this Section can be skipped.

The desired UR5 trajectory can be produced using `generateUR5Trajectory.m`. The trajectory is defined as a function within this script. Currently this function is defined as:

$$f(x) = \sqrt{0.15\cos(x)^2 - 1.5\sin(3x)^2 + 0.4\cos(2x)^2} \tag{C.1}$$

This function produces the trajectory illustrated in Figure C.1.

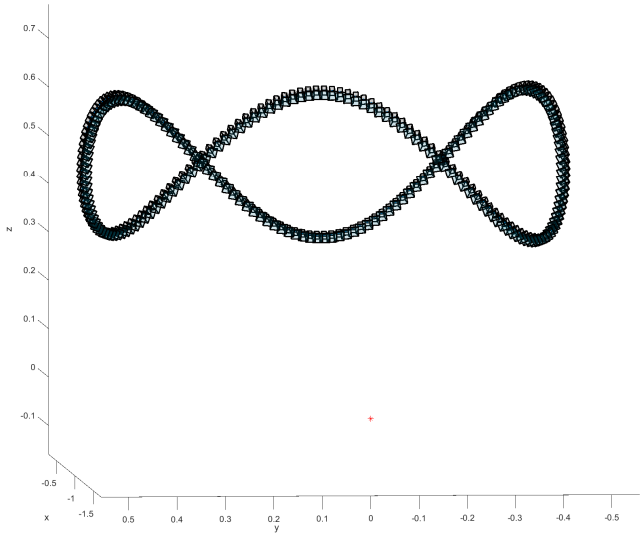


Figure C.1: Example trajectory of a camera attached to a UR5, with the centre of focus illustrated as a red dot [4].

Figure C.1 is generated by the same Matlab program, and indicates the progression of the focal centre of a camera attached to the UR5 as it moves through 200 posi-

tions. The figure also shows the location of the centre of focus as a red dot.

Running this `generateUR5Trajectory.m` produces `.SCRIPT` and `.TXT` files with the positions of the UR5's joints at points along the trajectory. The `.TXT` file is read by this software package, and the joint positions are then sent to the UR5 via Ethernet.

C.3.9 config.txt

An example `config.txt` file is provided below:

```
Lytro Remote Shutter Configuration File.

For a description of how this file works, please see the User Manual.

1. For 0.66fps video for a duration specified below enter '1',
For photos triggered by SPACEBAR enter '2',
For autonomous photos synchronised with a UR5 enter '3'
------------------------------------------------------------
3
------------------------------------------------------------

2. Do you want to download the photos to this computer once finished? (Y/N)
------------------------------------------------------------
N
------------------------------------------------------------

3. Do you want the photos to be deleted off the camera after they are
downloaded? (Y/N)
------------------------------------------------------------
N
------------------------------------------------------------

4. What format(s) would you like your photos to be saved as?
raw sensor .jpg ('j'), raw sensor .raw (r), raw sensor metadata ('m'),
stack of raw sensor images ('s'), processed .jpg ('J'),
processed .raw ('R'), processed text file ('T'),
processed light field photo (.lfp) ('L')
------------------------------------------------------------
L
------------------------------------------------------------
```

```
5. For Option 1, how long should the video be (in seconds)?
(leave as '0' if Option 1 is set to 2 or 3)
------------------------------------------------------------
0
------------------------------------------------------------

6. Perform UR5-Lytro calibration routine? (Y/N) (Option 1 must be set to '3')
------------------------------------------------------------
N
------------------------------------------------------------

7. Perform Lytro calibration routine using (Option 1 must be set to '1 or 2')
Format: (Y/N),[x,y],[x1,y1] (see User Manual for further details)
------------------------------------------------------------
N,[21 14],[12.2 12.2]
------------------------------------------------------------
```

C.4 Starting the Software

This section outlines how to run the software. This process involves the following steps:

1. Ensure that both Matlab and Lytro Desktop are installed on your PC (see Sections C.3.2 and C.3.3).
2. Ensure that you have previously connected the Lytro to your PC so that the white image data set could be downloaded (see Section C.3.2).
3. Alter and save the `config.txt` file as required (see Section C.3.9).
4. Connect the PC and UR5 via Ethernet cable.
5. Turn the UR5 on.
6. Ensure that the PC and UR5 are within the same IP subnet mask (see Section C.3.5).
7. Load the Polyscope program `Lytro.urp` onto the UR5 (don't run it) (see Section C.3.4).
8. Turn the Lytro camera on.
9. Enable the Lytro's wifi.
10. Connect your PC to the Lytro's wi-fi.

11. Start the software and follow the prompts.

C.5 UR5 Communication

Communication with the UR5 is done via Ethernet cable and TCP/IP protocol. See Section C.3.5 for an explanation of how this connection is set up. The communication is initiated and carried out using the Matlab functions `init()` and `moverobot()` [90]. The Lytro Remote Shutter software calls these functions by creating a reference to the Matlab Type Library [93]. The `init()` function opens the TCP/IP connection for the given IP address and port. The `moverobot()` function sends movement commands to the UR5 one at a time. These movement commands are extracted from the `.txt` file produced by the `generateUR5Trajectory.m` program described in Section C.3.8.

There is a 2 second delay between the sending of movement commands to the UR5. Half way through this time interval, the Lytro camera is triggered, thus ensuring that the UR5 has completed it's previous movement and is stationary when triggering occurs.

C.6 Downloading the Files

This software package allows images to be downloaded from the Lytro over wi-fi. The images can be downloaded in the following formats: `.raw, .jpg, .txt, .LFP, .JPG, .RAW`. The first three, lower-case file options refer to raw sensor data, while the upper-case options refer to data packages. Choosing which file types to download can be done in the `config.txt` file (see Section C.3.9). Any number of options can be chosen at one time. Ensure that multiple options are separated by commas in the `config.txt` file.

C.7 Matlab Processing

As mentioned in Section C.1, the processing of the light field files is done using the Matlab Light Field Toolbox created by Donald Dansereau. This toolbox is available here [64]. The required toolbox functions are run from the Matlab command line, which receives it's instructions from the Light Field Remote Shutter software.

The procedure that these instructions follow very closely mirrors the 'Quick Tour' that is described in the Light Field Toolbox user manual. Broadly, the procedure

undertakes the following steps:

1. Processing the White Images: A white image database is built using the white images specific to the Lytro camera being used. These files are loaded onto a PC when the Lytro is connected via USB. On Windows computers, these files are saved by default at the directory `<drive_letter>:\Users\<username>\AppData\Local\...Lytro\cameras\sn-<serial_number>`. These files are copied from this directory by this software package, and the white image database is then built using the Matlab toolbox.

2. Decoding the Light Field files: The light field images downloaded from the Lytro by this software package are decoded using the Matlab toolbox. For each image, the toolbox selects the corresponding white image, and saves the output files in the following formats: `*__Decode.mat` and `*.png`. These files are saved to the same directory that they are downloaded to.

3. Rectifying the Decoded files: In order to rectify the decoded light field files, the Lytro camera must first be calibrated. This can be done using the calibration data sets available here [94]. By default, the `PlenCalCVPR2013DatasetA` has been used. The result of this calibration is a `CalInfo.json` file, that contains information including pose, and intrinsic and distortion parameters. The lenslet grid model is also produced, an example of which is displayed below.

The decoded light field images are then rectified based on this calibration information, and the files are saved in the same location.

C.8 Accessing Decoded Files

The decoded and rectified light field images will be available at `<pwd>\LFToolbox0.4\...LFToolbox0.3_Samples1\Image\Lytro\` once the software has finished.

C.9 Directory Structure

```
LytroRemoteShutter
    RunDLL
        bin
            Release
                LFToolbox0.4
```

```
    LFToolbox0.3_Samples1
        Cameras
            A903200311
                01
                WhiteImages
        Images
            Lytro
    SupportFunctions
ur5_setup-master
    interface
UR5Trajectory

config.txt
Lytro Remote Shutter.exe
```

CMOS Parallel Pixel Data Interface

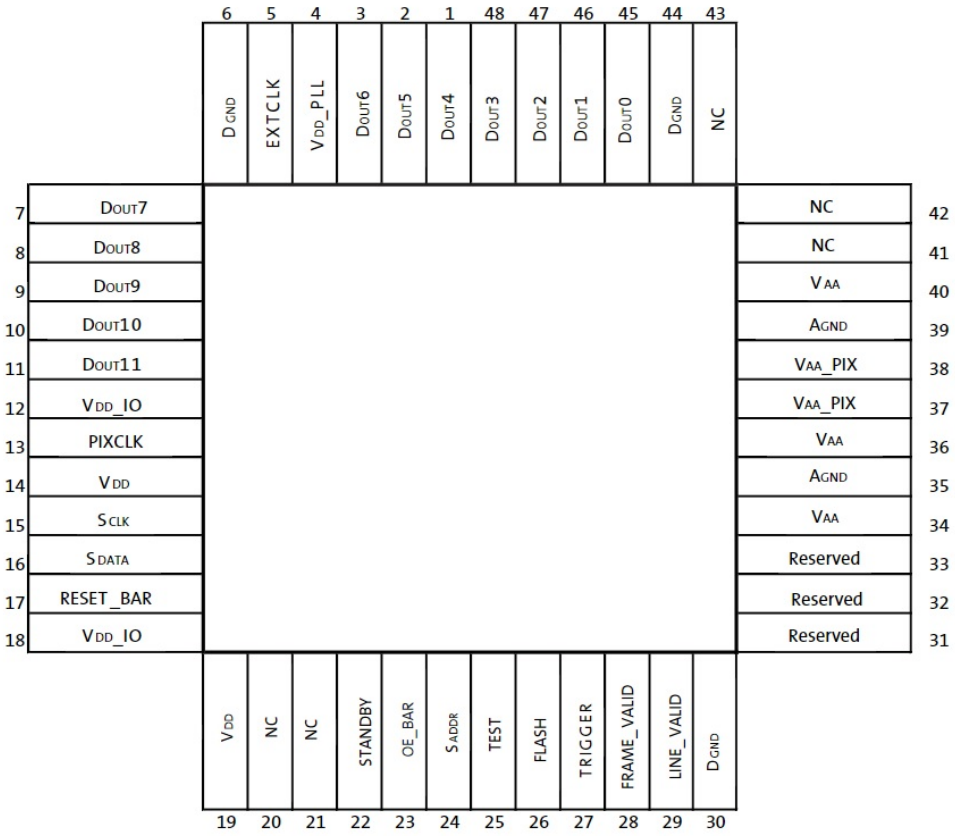


Figure D.1: Parallel Pixel Data interface pin diagram of similar Aptima CMOS [6].

Pin Number	Name	Type	Description
1	DOUT4	Output	Parallel pixel data output.
2	DOUT5	Output	Parallel pixel data output.
3	DOUT6	Output	Parallel pixel data output.
4	VDD_PLL	Power	PLL power.
5	EXTCLK	Input	External input clock.
6	DGND	Power	Digital ground.
7	DOUT7	Output	Parallel pixel data output.
8	DOUT8	Output	Parallel pixel data output.
9	DOUT9	Output	Parallel pixel data output.
10	DOUT10	Output	Parallel pixel data output.
11	DOUT11	Output	Parallel pixel data output (MSB).
12	VDD_IO	Power	I/O supply power.
13	PIXCLK	Output	Pixel clock out. DOUT is valid on rising edge of this clock.
14	VDD	Power	Digital power.
15	SCLK	Input	Two-Wire Serial clock input.
16	SDATA	I/O	Two-Wire Serial data I/O.
17	RESET_BAR	Input	Asynchronous reset (active LOW). All settings are restored to factory default.
18	VDD_IO	Power	I/O supply power.
19	VDD	Power	Digital power.
20	NC		No connection.
21	NC		No connection.
22	STANDBY	Input	Standby-mode enable pin (active HIGH).
23	OE_BAR	Input	Output enable (active LOW).
24	SADDR	Input	Two-Wire Serial address select.
25	TEST	Input	Manufacturing test enable pin (connect to DGND).
26	FLASH	Output	Flash output control.
27	TRIGGER	Input	Exposure synchronization input.
28	FRAME_VALID	Output	Asserted when DOUT frame data is valid.
29	LINE_VALID	Output	Asserted when DOUT line data is valid.
30	DGND	Power	Digital ground
31	Reserved	n/a	Reserved (do not connect).
32	Reserved	n/a	Reserved (do not connect).
33	Reserved	n/a	Reserved (do not connect).
34	VAA	Power	Analog power.
35	AGND	Power	Analog ground.
36	VAA	Power	Analog power.
37	VAA_PIX	Power	Pixel power.
38	VAA_PIX	Power	Pixel power.
39	AGND	Power	Analog ground.
40	VAA	Power	Analog power.
41	NC		No connection.
42	NC		No connection.
43	NC		No connection.
44	DGND	Power	Digital ground.
45	DOUT0	Output	Parallel pixel data output (LSB)
46	DOUT1	Output	Parallel pixel data output.

Figure D.2: Parallel Pixel Data interface signal descriptions [6].

Riser PCB Design

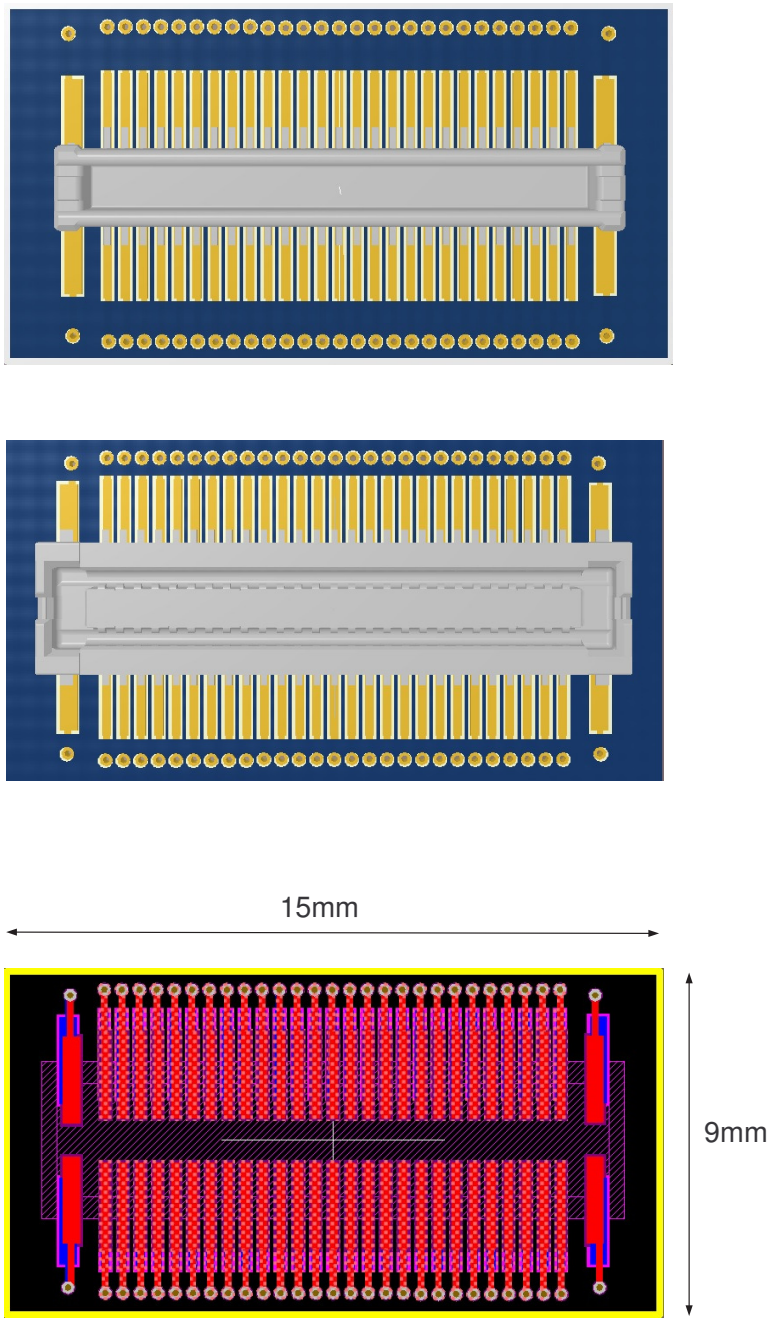


Figure E.1: Riser PCB designed in Altium Designer/ Bottom: 2D view, Middle: 3D view (back), Top: 3D view (front).

Table E.1: Riser PCB Layer Stack

Layer	**Name**	**Material**	**Thickness**	**Constant**
1	Top Overlay			
2	Top Solder	Solder Resist	0.015mm	3.5
3	Top Layer	Copper	0.051mm	
4	Dielectric 1	FR-4	0.381mm	4.2
5	Bottom Layer	Copper	0.051mm	
6	Bottom Solder	Solder Resist	0.152mm	3.5
7	Bottom Overlay			

Table E.2: Riser PCB component details

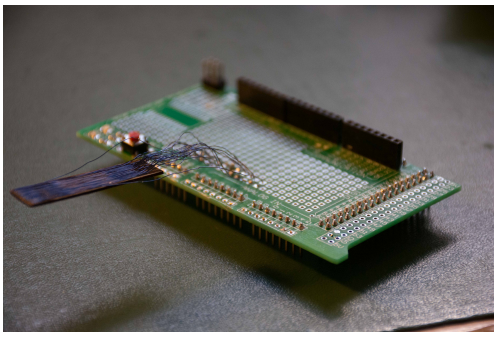
Board Component	**Amount on Board**	**Description**
54 Pin Mezzanine Connector (Header)	1	[67]
54 Pin Mezzanine Connector (Socket)	1	[67]
Pad Type 1	54	Length: 2mm, Width: 0.2mm, Solder Mask Expansion: 0.06mm
Pad Type 2	4	Length: 2mm, Width: 0.4mm, Solder Mask Expansion: 0.06mm
Trace	-	Width: 0.152mm
Via	58	Hole Size: 0.152mm, Hole Diameter: 0.305. Annular Ring width: 76.5mm, Solder Mask Expansion: 0.02mm

Form Tool Design

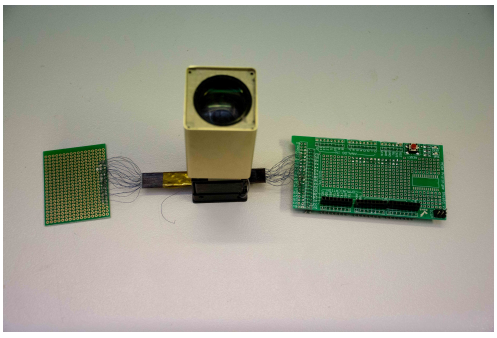
Four pieces of wood were cut and engraved with 27 grooves at 0.4mm spacing (so as to match the pitch of the connector on the PCB). These pieces are shown in Figure F.1a. The wood pieces were then paired, placed back-to-back, wrapped in wire and glued down, as shown in Figure F.1b. The wire was then cut along one end of the wood pieces, the wire was unwrapped and soldered to a prototyping PCB, as shown in Figure F.1c. The ends of the cut wire that were not soldered to the prototyping PCB were then soldered to the pins on the Riser PCB, to produce the final setup shown in Figure F.1d.

(a) Wood pieces with 0.4mm-spaced grooves and copper wire.

(b) Back-to-back wood pieces wrapped in wire.



(c) Cut wires soldered to a PCB.



(d) The Lytro Camera with completed Form Tool attached.

ZTEX USB-FPGA-Module 2.14

A list of features of the ZTEX USB-FPGA-Module 2.14 [8].

- USB 3.0 interface with Micro-USB connector
- Cypress CYUSB3033 EZ-USB FX3S Microcontroller:
  - ARM926EJ core at 208 MHz
  - 512 KByte SRAM
  - 1 Storage Port
  - 16 Bit GPIF II at 104 MHz (bandwidth between EZ-USB FX3S and FPGA: 204 MByte/s)
- Five Xilinx Artix 7 FPGA variants: XC7A15T, XC7A35T, XC7A50T, XC7A75T and XC7A100T, see Variants
- External I/O connector (consisting in two female 2x32 pin headers with 2.54mm grid) provides:
  - 100 General Purpose I/O's (GPIO) connected to FPGA
  - JTAG signals
  - Reset signal
  - External power (6 V .. 16 V) input
  - 3.3V output
  - I/O voltage output or input, see External I/O connector
  - Same pin assignment as USB-FPGA Module 2.13
- 256 MByte DDR3 SDRAM:
  - Up to 400 MHz clock frequency
  - 16 Bit bus width
  - Up to 1600 MByte/s data rate
  - Usable with the Xilinx Memory Interface Generator (MIG), examples are included in the SDK
- 128 MBit on-board Flash memory

  - Allows storage and auto-boot of Firmware
  - Allows storage and auto-boot of Bitstream
  - Accessible from EZ-USB FX3S and from FPGA, see the CPLD description

- SD card socket
  - SD 3.0 (SDXC) support
  - SDIO 3.0 support

- 2 Kbit MAC-EEPROM: contains a unique non erasable MAC-address and is used to store firmware settings
- On-Board power supply:
  - Two reverse current protected supply inputs: External power (6 V .. 16 V) and USB power
  - 5.0 V: 250 mA
  - 3.3 V: 3000 mA
  - 1.8 V: 1500 mA
  - 1.5 V: 1500 mA
  - 1.2 V; 1500 mA
  - 1.0 V; 4000 mA
  - OTG supply switchable and reverse current protected
- XC7A50T, XC7A75T and XC7A100T variants: Heat sink for high performance / high speed applications, see Section heat sink
- Optional:
  - Battery to store a key for bitstream encryption
- Temperature range: 0-70°C

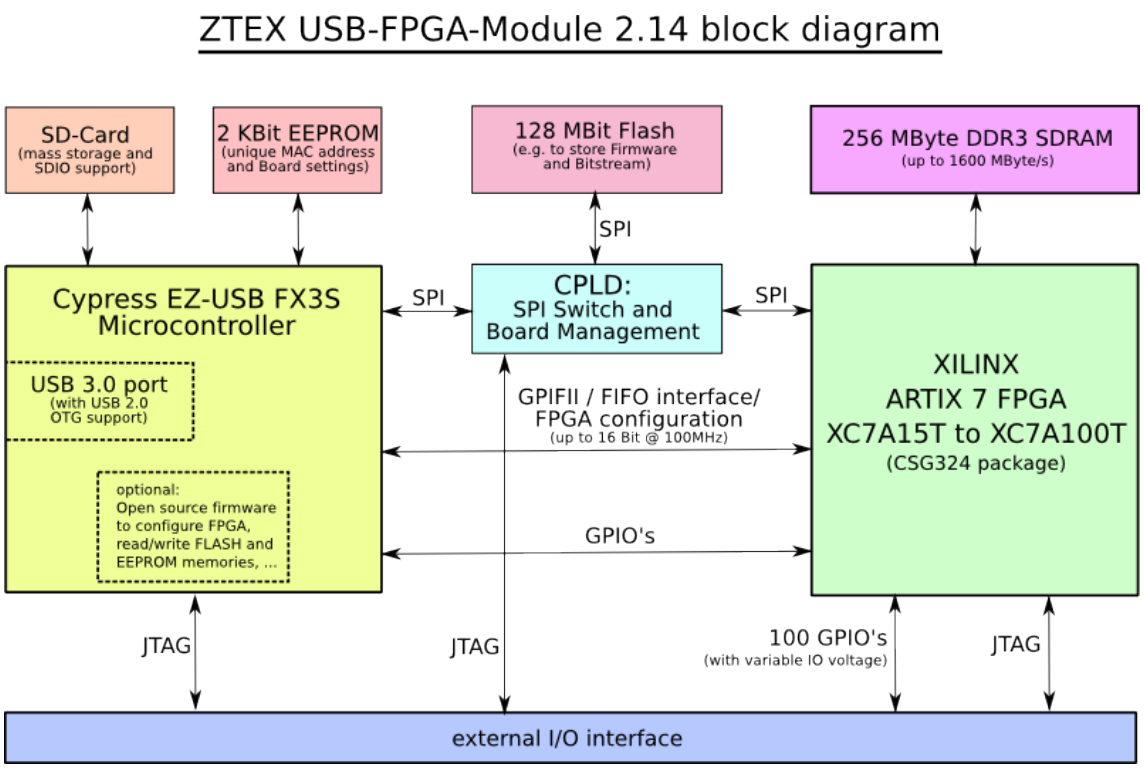


Figure G.1: ZTEX USB-FPGA-Module 2.14 Block Diagram [8].

Figure G.2: ZTEX USB-FPGA-Module 2.14 Back View [8].

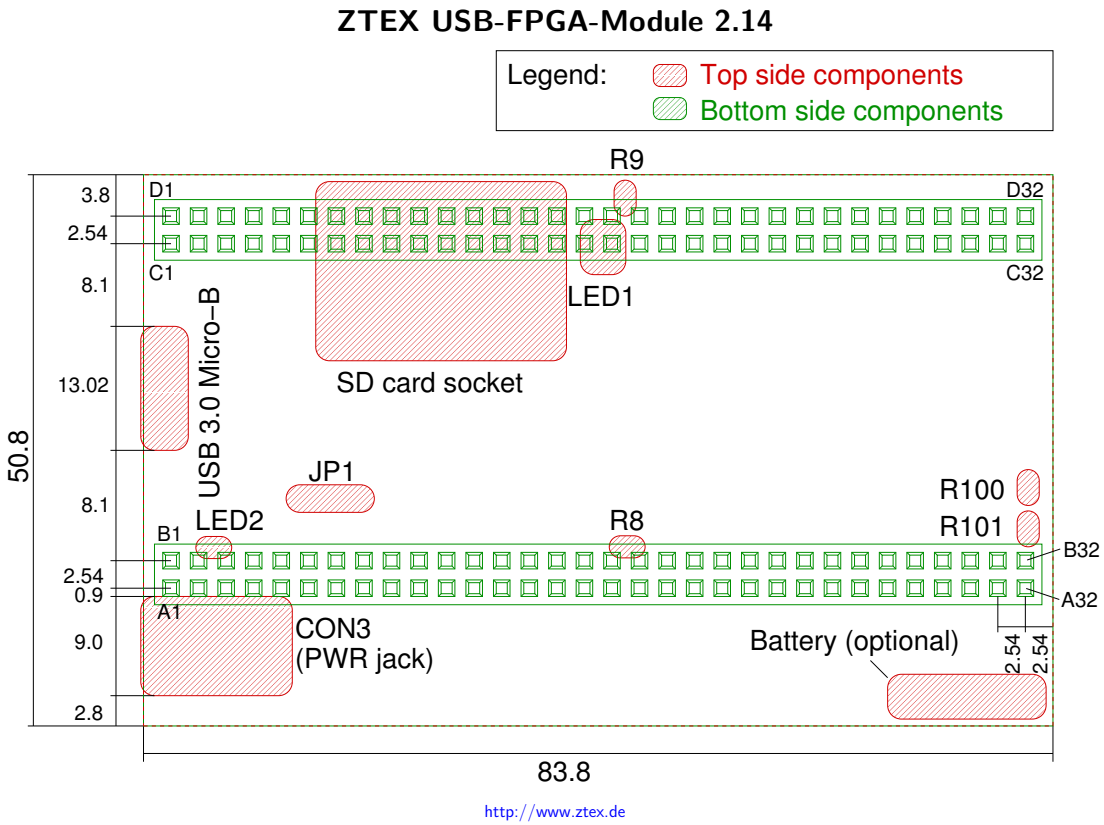


Figure G.3: ZTEX USB-FPGA-Module 2.14 Function Description [8].

Table G.1: ZTEX USB-FPGA-Module 2.14 Pin Assignment [8].

	Pinlist USB-Modules Series 2			
	A	**B**	**C**	**D**
1	VIN	VIN	USB+5V	Reset
2	GND	GND	GND	GND
3	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
4	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
5	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
6	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
7	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
8	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
9	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
10	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
11	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
12	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
13	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO

14	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
15	3.3V	3.3V	FPGA_IO	FPGA_IO
16	GND	GND	VCCO_CD	VCCO_CD
17	VCCO_AB	VCCO_AB	GND	GND
18	FPGA_IO	FPGA_IO	3.3V	3.3V
19	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
20	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
21	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
22	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
23	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
24	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
25	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
26	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
27	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
28	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
29	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
30	FPGA_IO	FPGA_IO	FPGA_IO	FPGA_IO
31	JTAG_TDI	JTAG_TCLK	JTAG_TDO	JTAG_TMS
32	JTAG_VIO	GND	GND	GND

Summary

	FPGA-IO	50	FPGA-IO	50
	3.3V	2	3.3V	2
	VIO_AB	2	VIO_CD	2
	JTAG	3	JTAG	2
	VIN	2	USB+5V	1
	GND	5	Reset	1
			GND	6
	Total:	64	Total:	64

Interceptor PCB Design

Table H.1: Interceptor PCB component details.

Board Component	Amount on Board	Description
54 Pin Mezzanine Connector (Header)	1	[67]
54 Pin Mezzanine Connector (Socket)	1	[67]
Pad Type 1	54	Length: 2mm, Width: 0.2mm, Solder Mask Expansion: 0.06mm
Pad Type 2	4	Length: 2mm, Width: 0.4mm, Solder Mask Expansion: 0.06mm
Trace	-	Width: 0.152mm
Via Type 1	58	Hole Size: 0.152mm, Via Diameter: 0.305. Annular Ring width: 0.0765mm, Solder Mask Expansion: 0.02mm
Via Type 2	64	Hole Size: 0.9mm, Via Diameter: 1.5mm. Annular Ring width: 0.6mm, Solder Mask Expansion: 0.02mm

The layer stack of the Interface PCB is the same as that of the Riser PCB, which is provided in Table E.1.

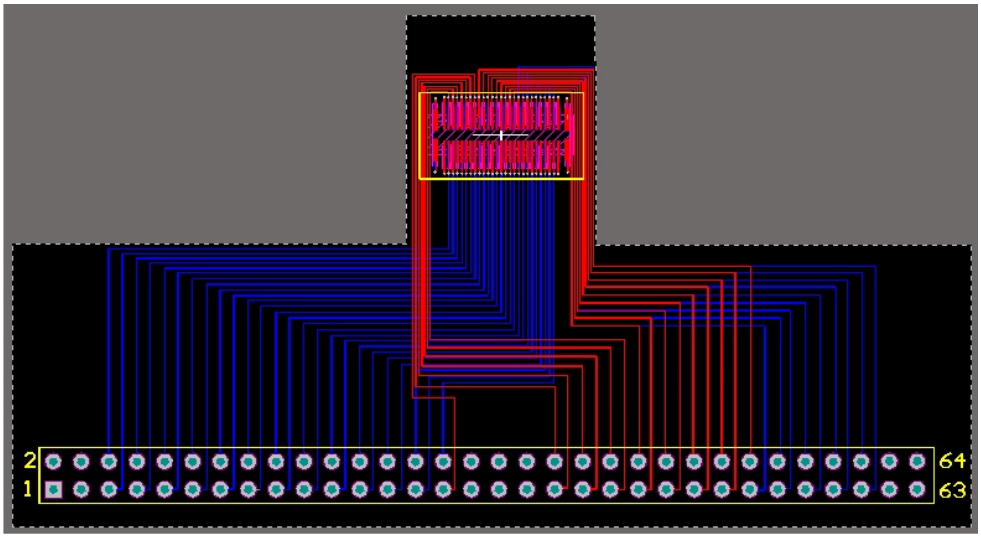


Figure H.1: 2D view of PCB designed to interface with ZTEX USB-FPGA-Module 2.14 (image created using Altium Designer software package).

Bibliography

[1] A. Software. Bayers filter. [Online]. Available: https://azzlsoft.com/tag/bayer-filter/

[2] J. Kucera, "Computational Photography of Light-field Camera and Application to Panoramic Photography," Master's thesis, Charles University, Prague, 2014.

[3] L. field forum. Lytro specifications: A deeper look inside. [Online]. Available: http://lightfield-forum.com/2012/11/lytro-specifications-a-deeper-look-inside/

[4] M. Abello, Personal Correspondence, 3 2017.

[5] O. Semiconductor. (2010) 1/2.3-inch 14 mp cmosdigital image sensor. [Online]. Available: http://www.onsemi.com/pub/Collateral/MT9F002-D.PDF

[6] Aptima. (2009) 1/3-inch cmos digital image sensor. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/APTINA%20PDFs/MT9M021,031.pdf

[7] Zoran. (2009) Coach-12v1080-bga / coach-12v60-bgacamera-on-a-chipdigital camera processorpreliminary data book. [Online]. Available: http://road-view.com/wp-content/uploads/2016/07/500F-Lense-Spec-COACH_12V1080_and_12V60_DataBook_v1.00.pdf

[8] ZTEX. (2017) Usb-fpga module 2.14: Artix 7 fpga board with ez-usb fx3s usb 3.0 controller and ddr3 sdram. [Online]. Available: https://www.ztex.de/usb-fpga-2/usb-fpga-2.14.e.html

[9] L. Jirkovský. Lytro protocol. [Online]. Available: https://ljirkovsky.wordpress.com/2015/03/16/lytro-protocol/

[10] U. I. Forum. Universal serial busmass storage class bulk-only transportrevision. [Online]. Available: http://www.usb.org/developers/docs/devclass_docs/usbmassbulk_10.pdf

[11] J. Kucera. (2014) Lytro meltdown. [Online]. Available: http://optics.miloush.net/lytro/

[12] K. Ahonen. Transport control protocol. [Online]. Available: https://www.netlab.tkk.fi/opetus/s38130/s98/tcp/TCP.htm

[13] K. J, Personal Communication, 7 2017.

[14] G. Kennedy. (2017) Lytro remote shutter. [Online]. Available: https://github.com/kennege/Lytro-Remote-Shutter

[15] S. Cooper. What is light field? [Online]. Available: http://blog.lytro.com/what-is-light-field/

[16] W. J. Adelson E, "Single lens stereo with a plenoptic camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 99–106, 1992.

[17] B. M. D. G. H. M. H. P. Ng R, Levoy M, "Light field photography with a hand-held plenoptic camera," *Technical Report, Stanford University*, 2005.

[18] R. Ng, "Digital light field photography," Ph.D. dissertation, Stanford University, 2006.

[19] D. D, "Plenoptic signal processing for robust vision in field robotics," Ph.D. dissertation, Sydney University, 2014.

[20] G. Wu, B. Masia, A. Jarabo, Y. Zhang, L. Wang, Q. Dai, T. Chai, and Y. Liu, "Light field image processing: An overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1–1, 2017.

[21] S. X. E.-C. R. B. R. Dong F, Ieng S, "Plenoptic cameras in real-time robotics," *The International Journal of Robotics Research*, vol. 32(2), pp. 206–217, 2013.

[22] Lytro. Lytro user manual. [Online]. Available: http://lytro-s3-production.s3.amazonaws.com/media/lytro/Official-User-Guide.pdf

[23] B. J. Adelson E, "The plenoptic function and the elements of early vision," *Computational Models of Visual Processing*, pp. 3–20, 1991.

[24] H. P. Levoy M, "Light field rendering," *ACM*, pp. 31–42, 1996.

[25] T. Georgiev, Z. Yu, and G. S. Lumsdaine, Andrew, "Lytro camera technology: theory, algorithms, performance analysis," *Proc. SPIE*, vol. 8667, pp. 86 671J–86 671J–10, 2013. [Online]. Available: http://dx.doi.org/10.1117/12.2013581

[26] M. G. Lippmann, "Reversible prints. integral photographs," *J. Phys.*, vol. 7, pp. 821–825, 1908, translated by Durand F, MIT CSAIL.

[27] T. K. Ishihara Y, "A high photosensitivity il-ccd image sensor with monolithic resin lens array," *In proceedings of IEEE IEDM*, pp. 497–500, 1983.

[28] S. Ogata, J. Ishida, and T. Sasano, "Optical sensor array in an artificial compound eye," *Optical Engineering*, vol. 33, no. 11, pp. 3649–3655, 1994. [Online]. Available: http://dx.doi.org/10.1117/12.179889

[29] E. R. Dowski, Jr. and G. E. Johnson, "Wavefront coding: a modern method of achieving high-performance and/or low-cost imaging systems," *Proc. SPIE*, vol. 3779, pp. 137–145, 1999. [Online]. Available: http://dx.doi.org/10.1117/12.368203

[30] L. F. Forum. (2017) Raytrix r11 3d lightfield camera. [Online]. Available: http://lightfield-forum.com/raytrix/raytrix-r11-3d-lightfield-camera/

[31] L. Inc. (2017) Technical specifications. [Online]. Available: https://illum.lytro.com/illum/specs

[32] L. F. Forum. (2017) Raytrix r5 high-speed video light-field camera. [Online]. Available: http://lightfield-forum.com/raytrix/raytrix-r5-high-speed-video-lightfield-camera/

[33] ——. (2017) Raytrix r29 dual-gige lightfield camera. [Online]. Available: http://lightfield-forum.com/raytrix/raytrix-r29-dual-gige-lightfield-camera/

[34] ——. (2017) Pelican imaging array camera: Light field module for smartphones. [Online]. Available: http://lightfield-forum.com/light-field-camera-prototypes/pelican-imaging-array-camera-light-field-module-for-smartphones/

[35] A. J. (2013) Toshiba shows off 'lytro chips' that convert phones to light-field cameras. [Online]. Available: http://www.computerworld.com/article/2495589/computer-processors/toshiba-shows-off--lytro-chips--that-convert-phones-to-light-field-cameras.html

[36] H. M. (2015) Apple's linx purchase: Dslr-killer or just long-term planning? [Online]. Available: https://www.extremetech.com/extreme/203433-apples-linx-purchase-dslr-killer-or-just-long-term-planning

[37] S. S. Snoswell A, "Light field de-blurring for robotics applications," *Australasian Conference on Robotics and Automation*, 2013.

[38] P. O. W. S. Dansereau D, Mahon I, "Plenoptic flow: Closed-form visual odometry for light field cameras," *Intilligent Robots and Systems (IROS), IEEE/RSJ Intl. Conf. on*, pp. 4455–4462, 2011.

[39] D. Dansereau and L. Bruton, "Gradient-based depth estimation from 4d light fields," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 3, pp. III – 549, 06 2004.

[40] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi, "Depth from combining defocus and correspondence using light-field cameras," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ser. ICCV '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 673–680. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2013.89

[41] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk, "Light field mapping: Efficient representation and hardware rendering of surface light fields," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 447–456, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566601

[42] C. Kim, "3d reconstruction and rendering from high resolution light fields," Ph.D. dissertation, ETH Zurich, 2015.

[43] T. E. Bishop, S. Zanetti, and P. Favaro, "Light field superresolution," pp. 1–9, April 2009.

[44] M. Abello, "Rigid body state estimation from sparse range measurements," *Undergraduate Research Project, Australian National University*, 2016.

[45] X. Yu, R. Wang, and J. Yu, "Real-time depth of field rendering via dynamic light field generation and filtering," *Computer Graphics Forum*, vol. 29, no. 7, pp. 2099–2107, 2010. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2010.01797.x

[46] A. Davis, M. Levoy, and F. Durand, "Unstructured light fields," *Computer Graphics Forum*, vol. 31, no. 2pt1, pp. 305–314, 2012. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2012.03009.x

[47] M. Archambault. (2017) The science behind lytro's light field technology and megaray sensors. [Online]. Available: https://petapixel.com/2015/06/22/the-science-behind-lytros-light-field-technology-and-megaray-sensors/

[48] D. Murray and J. J. Little, "Using real-time stereo vision for mobile robot navigation," *Autonomous Robots*, vol. 8, no. 2, pp. 161–171, Apr 2000. [Online]. Available: https://doi.org/10.1023/A:1008987612352

[49] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," vol. 3, pp. 1063–1068, 2006.

[50] Y. Matsumoto and A. Zelinsky, "An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement," pp. 499–504, 2000.

[51] M. Bertozzi and A. Broggi, "Gold: a parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Transactions on Image Processing*, vol. 7, no. 1, pp. 62–81, Jan 1998.

[52] U. Franke and A. Joos, "Real-time stereo vision for urban traffic scene understanding," pp. 273–278, 2000.

[53] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, "A stereo machine for video-rate dense depth mapping and its new applications," pp. 196–202, June 1996.

[54] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," vol. 1, pp. I–195–I–202 vol.1, June 2003.

[55] M. Kuhn, S. Moser, O. Isler, F. K. Gurkaynak, A. Burg, N. Felber, H. Kaeslin, and W. Fichtner, "Efficient asic implementation of a real-time depth mapping stereo vision system," vol. 3, pp. 1478–1481 Vol. 3, Dec 2003.

[56] L. F. Forum. (2017) Lytro lightfield camera. [Online]. Available: http://lightfield-forum.com/lytro/lytro-lightfield-camera/

[57] Ebay. (2017) Lytro digital cameras. [Online]. Available: https://www.ebay.com.au/b/Lytro-Digital-Cameras/31388/bn_25133808

[58] Raytrix. (2017) Products & reseller prices. [Online]. Available: http://www.isolutions.com.sg/Raytrix.pdf

[59] Lytro. Lytro deskop app. [Online]. Available: https://illum.lytro.com/desktop

[60] L. Jirkovsky. Lyli. [Online]. Available: https://bitbucket.org/stativ/lyli

[61] M. Avastar. Marvell avastar 88w8787 soc. [Online]. Available: http://www.marvell.com.cn/wireless/assets/8787.pdf

[62] MathWorks. Downloads. [Online]. Available: https://au.mathworks.com/downloads/

[63] U. Robots. (2014) User manual. [Online]. Available: https://www.universal-robots.com/media/8704/ur5_user_manual_gb.pdf

[64] D. Dansereau. Light field toolbox v0.4. [Online]. Available: https://au.mathworks.com/matlabcentral/fileexchange/49683-light-field-toolbox-v0-4?requestedDomain=www.mathworks.com

[65] ——, Personal Correspondence, 10 2017.

[66] R. Hartley, Personal Correspondence, 10 2017.

[67] digikey. Panasonic f4s connector. [Online]. Available: https://www.digikey.com/products/en/connectors-interconnects/rectangular-board-to-board-connectors-arrays-edge-type-mezzanine/308?FV=ffec4e572Cffe00134&mnonly=0&newproducts=0&ColumnSort=1000002&page=6&stock=0&pbfree=0&rohs=0&quantity=&ptm=0&fid=0&pageSize=25

[68] USB.org. (2017) Usb 3.1 specification. [Online]. Available: http://www.usb.org/developers/ssusb/USB_3_1_Language_Product_and_Packaging_Guidelines_FINAL.pdf

[69] J. Andrew. (2017) Wifi transfer rates types. [Online]. Available: https://itstillworks.com/wifi-transfer-rates-types-21881.html

[70] T. Instruments. (2017) Omap-l138 c600 dsp+ arm processor datasheet. [Online]. Available: http://www.ti.com/lit/ds/symlink/omap-l138.pdf

[71] Milbeaut. (2017) Image-processing system lsi for digital cameras milbeaut m-4 series mb91680. [Online]. Available: https://www.fujitsu.com/downloads/EDG/binary/pdf/find/24-1e/3.pdf

[72] Panasonic. (2017) Mn101 series mn103 series & development environments. [Online]. Available: http://www.aldebaran.cz/bulletin/2014_10/catalog_A00013E.pdf

[73] O. Semiconductor. (2017) Image processors. [Online]. Available: http://www.onsemi.com/PowerSolutions/parametrics.do?id=16880

[74] Sansung. (2017) Samsung exynos 5 dual (exynos 5250) risc microprocessor. [Online]. Available: http://www.arndaleboard.org/wiki/downloads/supports/Exynos_5_Dual_User_Manaul_Public_REV1.00.pdf

[75] ——. (2017) Dsp - digital signal processing design. [Online]. Available: https://www.xilinx.com/training/dsp/dsp-digital-signal-processing.htm

[76] G. R. Goslin, "Guide to using field programmable gate arrays (fpgas) for application-specific digital signal processing performance," *Proc.SPIE*, vol. 2914, pp. 2914 – 2914 – 11, 1996. [Online]. Available: http://dx.doi.org/10.1117/12.255830

[77] R. Wimalagunarathne, C. Wijenayake, A. Madanayake, D. G. Dansereau, and L. T. Bruton, "Integral form 4-d light field filters using xilinx fpgas and 45 nm cmos technology," *Multidimensional Systems and Signal Processing*, vol. 26, no. 1, pp. 47–65, Jan 2015. [Online]. Available: https://doi.org/10.1007/s11045-013-0235-6

[78] Y. Martin, L. F. Rodriguez-Ramos, J. García, J. J. D. García, and J. M. Rodriguez-Ramos, "Fpga-based real time processing of the plenoptic wavefront sensor for the european solar telescope (est)," pp. 87–92, March 2010.

[79] R. J. Petersen and B. L. Hutchings, *An assessment of the suitability of FPGA-based systems for use in digital signal processing.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 293–302. [Online]. Available: https://doi.org/10.1007/3-540-60294-1_123

[80] H.-H. K. L. M. A. H. Bennett S. Wilburn, Michal Smulski, "Light field video camera," *Proc.SPIE*, vol. 4674, pp. 4674 – 4674 – 8, 2001. [Online]. Available: http://dx.doi.org/10.1117/12.451074

[81] C. Desmouliers, E. Oruklu, S. Aslan, J. Saniie, and F. M. Vallina, "Image and video processing platform for field programmable gate arrays using a high-level synthesis," *IET Computers Digital Techniques*, vol. 6, no. 6, pp. 414–425, November 2012.

[82] H. Afshari, A. Akin, V. Popovic, A. Schmid, and Y. Leblebici, "Real-time fpga implementation of linear blending vision reconstruction algorithm using a spherical light field camera," pp. 49–54, Oct 2012.

[83] P. Greisen, S. Heinzle, M. Gross, and A. P. Burg, "An fpga-based processing pipeline for high-definition stereo video," *EURASIP Journal on Image and Video Processing*, vol. 2011, no. 1, p. 18, Nov 2011. [Online]. Available: https://doi.org/10.1186/1687-5281-2011-18

[84] F. Roth, "Using low cost FPGAs for realtime video processing," Master's thesis, Masaryk University, Brno, 2011.

[85] Xilinx. (2017) Artix-7. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html

[86] ——. (2017) All programmable 7 series product selection guide. [Online]. Available: https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf

[87] ZTEX. (2017) Trace length in mm. [Online]. Available: https://www.ztex.de/downloads/usb-fpga-2.14-io_traces.txt

[88] Modtronix. (2017) 2x32 pin header, 2.54mm. [Online]. Available: http://modtronix.com/hdr2x32-m254.html

[89] PCBCART. (2017) Pcbcart. [Online]. Available: https://www.pcbcart.com/

[90] F. Aalamifar. Ur5 control using matlab. [Online]. Available: https://au.mathworks.com/matlabcentral/fileexchange/50655-ur5-control-using-matlab

[91] L. Qian. Ur5 complete setup guide. [Online]. Available: https://github.com/qian256/ur5_setup

[92] W. S. Dansereau D, Pizarro O, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," *Computer Vision and Pattern Recognition (CVPR), IEEE Conf. on*, pp. 1027–1034, 2013.

[93] MathWorks. Pass complex data to matlab from c# client. [Online]. Available: https://au.mathworks.com/help/matlab/matlab_external/call-matlab-function-from-a-c-client.html

[94] D. Dansereau. Plenoptic imaging. [Online]. Available: http://marine.acfr.usyd.edu.au/research/plenoptic-imaging/