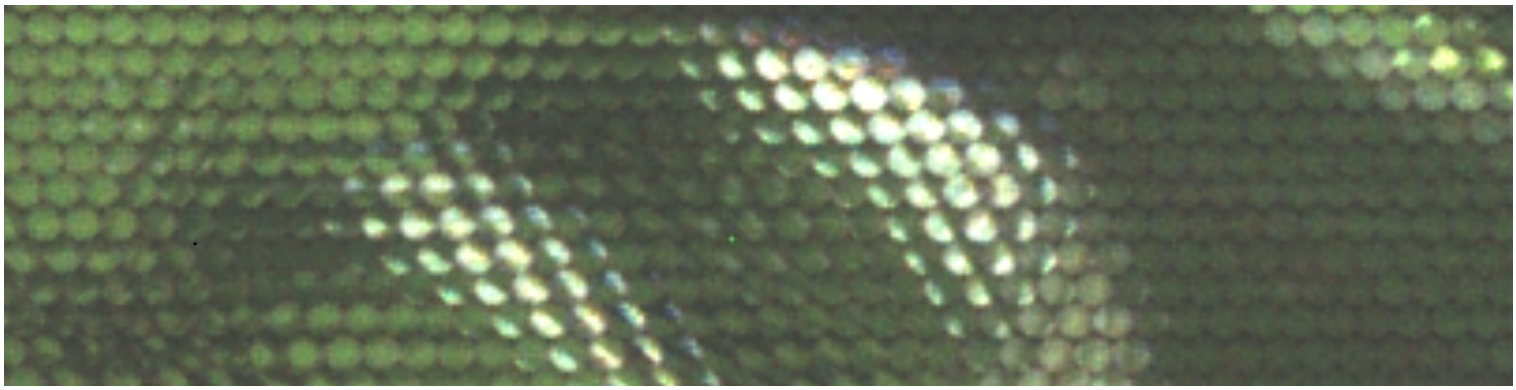


Depth scene estimation from images captured with a plenoptic camera

Interneship Report

Author
Gustavo Luiz SANDRI

Tutor (ENSEIRB)
M. Yannick BERTHOUMIEU

Tutors (Technicolor)
Mme. Neus SABATER
M. Valter DRAZIC

February to July 2013

Contents

Acknowledgment

I would like to extend gratitude to my supervisors, Valter Drazic and Neus Sabater, who afforded me the opportunity to intern at Technicolor. I want to deeply thank them for their remarkable supervision of my internship. Their support, their kindness and their assistance in writing this report have been a great help during all my work here.

I want to express my thanks and my friendliness towards everyone at the work package I were inserted for providing me with a really friendly and enjoyable working environment during all those months, as well to all the others interns for the environment created.

To my teachers at the Université Bordeaux I and ENSEIRB-MATMECA for the knowledge they transmitted and specially to Marc Donias, Jean-François Giovannelli, Nathalie Deltimple, Eric Kerhervé and Yannick Berthoumieu for understanding my needs and making this possible.

And least, but not less import, I would like to express my gratitude to my friends and family for sharing the good and bad times, specially to Transílita Sandri to whom I can unconditionally trust.

To Phoenix

1 Abstract

A plenoptic camera, also known as light field camera, is a device that employs a microlens array placed between the main lens and the camera sensor to capture the 4D light field information about a scene. Such light field enable us to know the position and angle of incidence of the light rays captured by the camera and can be used to improve the solution of computer graphics and computer vision-related problems.

With a sampled light field acquired from a plenoptic camera, several low-resolution views of the scene are available from which to infer depth. Unlike traditional multiview stereo, these views are captured by the same sensor, implying that they are acquired with the same camera parameters. Also the views are in perfect epipolar geometry.

However, other problems arises with such configuration. The camera sensor uses a Bayer color filter and demosaicing the RAW data implies view cross-talk creating image artifacts. The rendering of the views modify the color pattern, adding complexity for demosaicing.

The resolution of the views we can get is another problem. As the angular and spatial position of the light rays are sampled by the same sensor, there is a trade off between view resolution and number of available views. For Lytro camera, for example, the views are rendered with about 0.12 megapixels of resolution, implying in aliasing on the views for most of the scenes.

This work present: an approach to render the views from the RAW image captured by the camera; a method of disparity estimation adapted to plenoptic cameras that enables the estimation even without executing the demosaicing; a new concept of representing the disparity information on the case of multi-view stereo; a reconstruction and demosaicing scheme using the disparity information and the pixels of neighbouring views.

2 Introduction

Leonard da Vinci [24], on his manuscript about painting, described the nature of light by an infinite number of radiant pyramids trying to represent the light emanating of an object: "Every body in the light and shade fills the surrounding air with infinite images of itself", as described by the Figure 1. This explains the image formation with a pinhole camera as the effect of selecting one of this pyramids and projecting a cone of light on a sheet of paper [1].

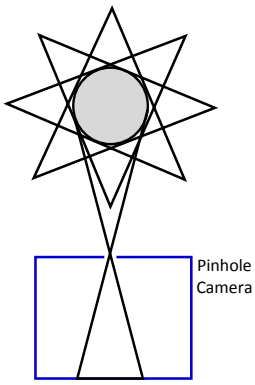


Figure 1: Figure based on a diagram from Leonardo da Vinci's notebooks illustrating the light emanating from the object in structures he called pyramids. A pinhole camera captures one of this pyramids to form an image

Nowadays, with the evolution of the light theory, we prefer to refer to image formation in terms of light rays rather than the poetic Leonard's "radiant pyramids" [18].

Mathematically speaking, the complete light information of an environment can be expressed by a plenoptic function, which describes the radiation of the totality of light rays filling a given region of space at any moment. It gives the radiation of a light ray, crossing the point (x, y, z), with an incoming angle of (θ_x, θ_y) at a given moment t. So, the plenoptic function has six dimensions.

In the case of photography we can ignore the temporal dimension and work with the instantaneous plenoptic function. Moreover we can eliminate one spatial dimension by analysing only the light rays crossing a given plane. As long as the rays are not blocked on their way travelling the space, one can predict the configuration of the light rays in another plan just by propagation. This resulting four dimension function seems to be well suited to describe a complete scene and is called light field.

The problem at this point becomes how to capture the light field. If we consider a pinhole camera we can observe that the function of the hole is to select light rays crossing a given position of the space (the hole position) and the resulting rays fall on the sheet of paper according to their incident angle. Most of the information is lost.

Conventional cameras use a lens instead of a pinhole, capturing a continuum of positions, imaging them at the sensor plane. This results in more light entering the device, reducing the noise effect. But the

sensor works as an integration device, combining together all light rays falling on the same sensor pixel losing the directional information of each ray. Therefore, just a small portion of the light field, present on a two-dimensional photo, is captured.

One simple idea to capture more information is to use two cameras, placed at different positions, capturing at the same time the light of a scene [39]. The image of the two cameras together offers two samples of the light field and this richer information enable us to estimate the missing parts of the light field and gather information about the depth of the scene. We could improve the quality of the measure using even more cameras [17]. The main draw back of this framework is that the cameras need to be synchronized, precisely disposed to obtain an epipolar geometry and have the same calibration.

We can avoid the use of two, or more cameras, by taking photos of different views with the same camera, either by placing the camera in several positions to take the photo, what requires precise displacement and a still scene, or by selecting portions of the main lens with a programmable aperture, as proposed in [25], which offers information about several views of the scene similar to the ones captured by a pinhole camera, when its "hole" is displaced to positions restricted to the lens surface.

Another form of capturing the light field arises from a modification of a conventional camera. The aim is to avoid the integration on the sensor plane, as proposed by Gabriel Lippmann in 1908 [26], which he called "the integral camera". An array of microlenses, as in Figure 2, is placed before the sensor deviating the rays hitting each microlens as a function of its incident angle. As a consequence, the position where the light arrive to the sensor plane gives information about the spacial position of the ray (which microlens it crossed) and its angle of incidence.

The aim of Lippmann was to use this camera to create tridimensional images that could be visualized using an identical device, generating a 2D image on the sensor plane and using the microlens array to rearrange the rays to compose a 3D image.

But Lippmann's idea was not technologically doable at that time. Three years later, the Russian physicist P. P. Sokolov, modified the concept in order to construct the first integral camera, replacing the microlenses by pinholes.

During the last century, the idea developed [7, 9] with the same principle, but it was just with the advent of digital processing that this technique could finally be fully exploited and became a practical tool [29, 27], making it possible to perform more complex processing, as refocusing [28, 21, 35] and depth estimation [4, 5].

Commercially, the first plenoptic cameras to be launched in the market is produced by Raytrix [36], devoted to research applications. For the general customer, Lytro, a Silicon Valley start up, sells the first consumer light-field camera.

The Lytro camera uses a glass plate containing thousands of microlenses, which are positioned between a main zoom lens and a standard 11-megapixel digital image sensor. The main lens images the subject onto the microlenses. Each microlens in turn focus on the main lens, which from the perspective of a microlens is at optical infinity.

Light rays arriving at the main lens from different angles are focused onto different microlenses. Each microlens in turn projects a tiny blurred image onto the sensor behind it. In this way the sensor records both the position of each ray as it passes through the main lens and its angle in a single exposure.

In this work we are going to concentrate on the information provided by the Lytro camera to estimate the depth of the objects in the scene where the photo was shot.

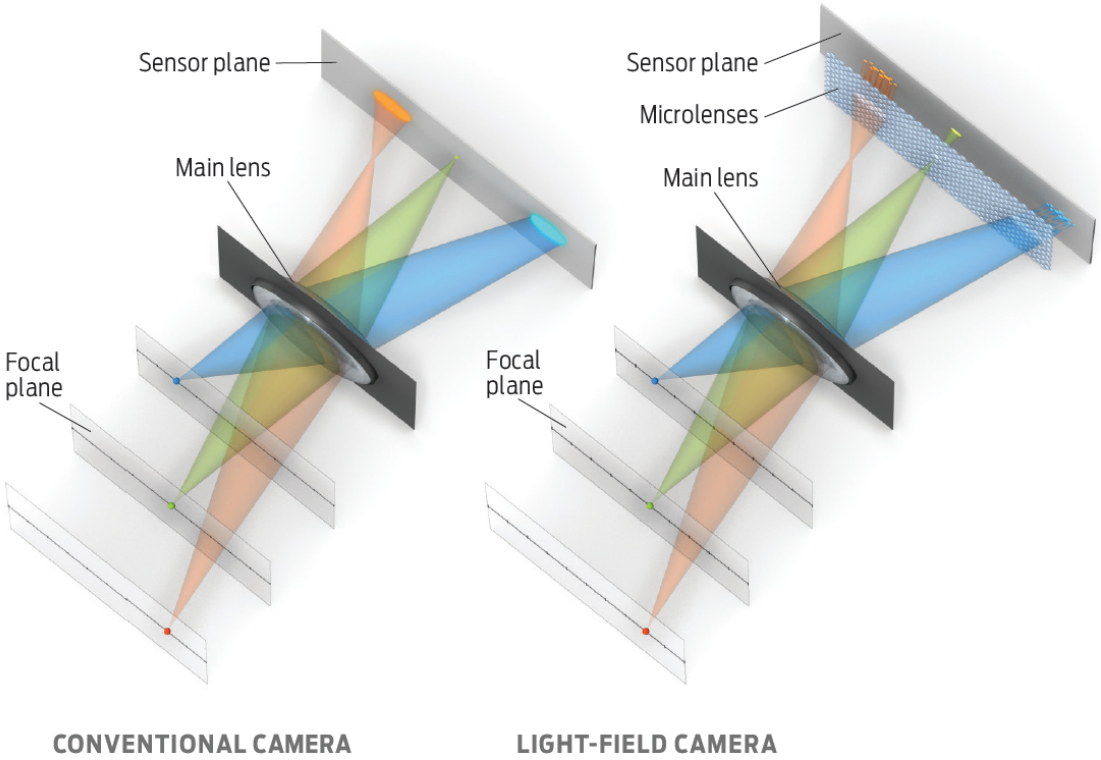


Figure 2: Conventional versus plenoptic camera

3 Working Environment

Technicolor SA, formely Thomson SA and Thomson Multimedia, with more than 95 years of experience in entertainment innovation, serves as international base of entertainment, software, and gaming customers. The company is a leading provider of production, post-production, and distribution services to content creators, network service providers and broadcasters.

Technicolor is one of the world's largest film processors; the largest independent manufacturer and distributor of DVDs (including Blu-ray Disc); and a leading global supplier of set-top boxes and gateways. The company also operates an Intellectual Property and Licensing business unit.

Technicolor's headquarters is located in Issy les Moulineaux - France. Other main office locations include Rennes (France), Edegem (Belgium), Wilmington (Ohio, USA), Burbank (California, USA), Princeton (New Jersey, USA), London (England, UK), Rome (Italy), Madrid (Spain), Hilversum (Netherlands), Bangalore (India) and Beijing (China).

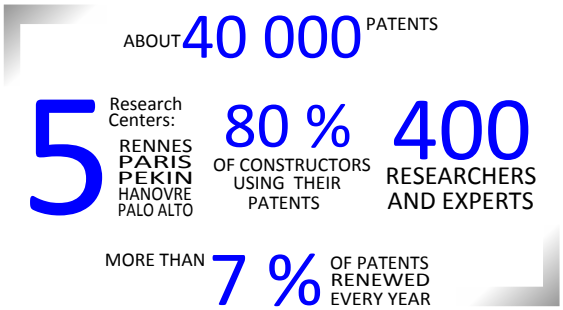
The activities developed at Tecnicolor can be organized into three operating divisions, as shown below:

TECHNOLOGY
- Research & Innovation
- Licensing
- Media Navi

ENTERTAINMENT SERVICES
- Creative Services (visual effects, animation & post production)
- Theatrical Services (photochemical film and digital cinema production)
- DVD & Blu-ray™ Services

DIGITAL DELIVERY
- Connected Home

The main clients of Technicolor are:



- **Telephone operating companies**: Comcast, UPC, Telefonica, BT, France Télécom, Orange, Tata Sky, Sky, DIRECTV, AT & T, StarHub, SK telecom, Verzion, Astro
- **Audiovisual Communication (TV)**: Canal+, TF1, France 24, ABC, Fox, CBS, itv, TV5 Monde, BBC Worldwide, HBO, CCTV
- **Film production**: Media consortium Disney, News Corporation, Viacom, NBC Universal, DreamWorks, TimeWarner, Sony
- **Innovative video users**: Wal-Mart, Circuit City, Darty, Carrefour, Best Buy, Costco, FedEx, EA, Microsoft

3.1 Technicolor Rennes

Technicolor Rennes is the largest technology centre of Technicolor Group. Pole of excellence in the whole image chain, the Rennes centre is the supplier of innovative solutions for Media & Entertainment Industries. A part of the activities is focused on the compression, protection, transmission, production and management of high definition contents while other technological solutions are developed to respond to the demands of network operators and to meet the needs of Media & Entertainment customers.

Technicolor Rennes has built partnerships with top academic institutes in Europe, public institutes, industrial partners and is involved in all French and European research programs such as RNRT, RNTL, RIAM, MEDEA, ITEA, IST, CNES, ESA, among others.

The effectif at Rennes is of 550 employees, where 202 are researchers and experts, representing the biggest concentration of researchers at the group.

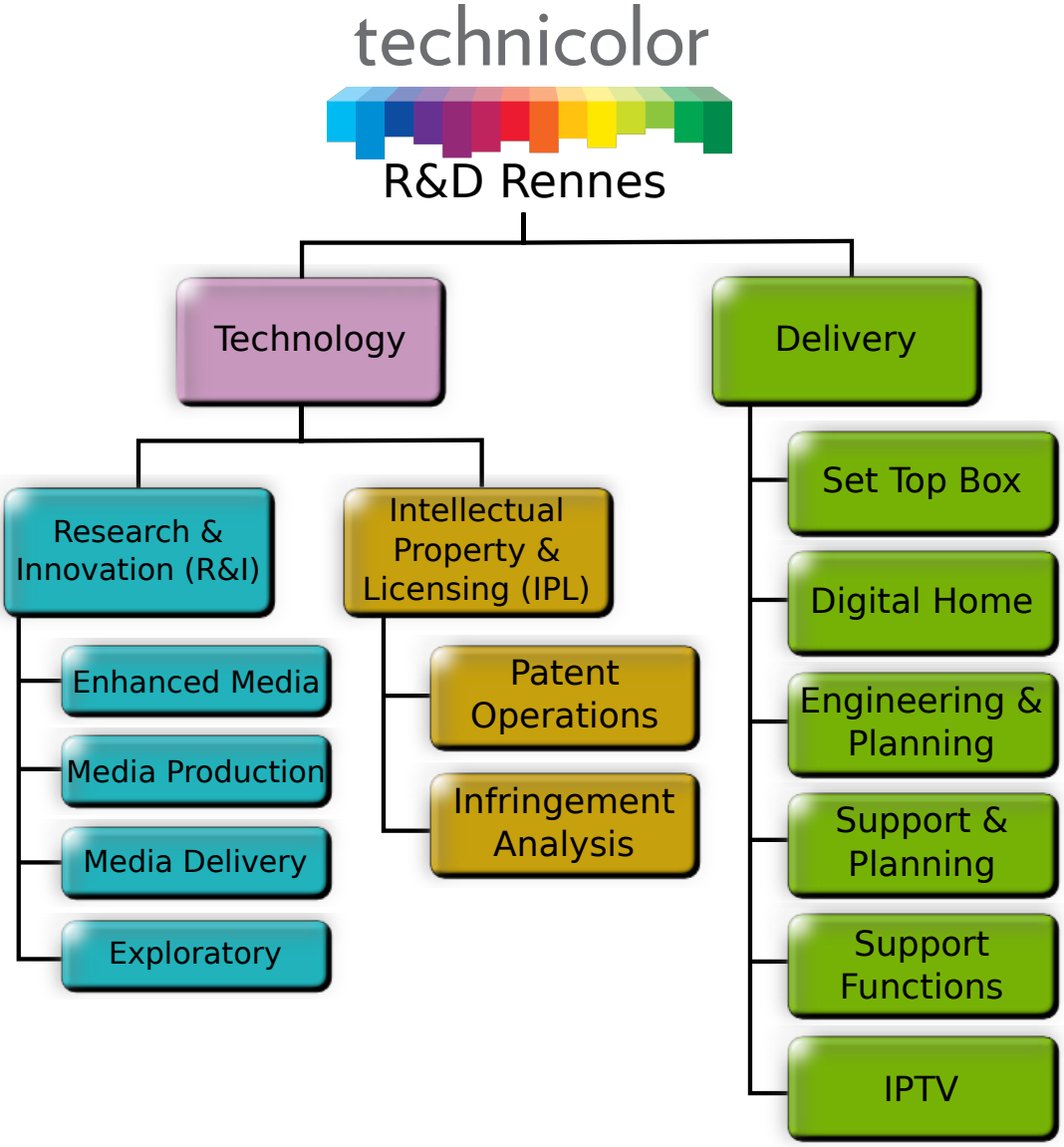
The center is divided into two branches:

- **Tecnology** containing the pole Research & Innovation (R&I) and Intellectual Property & Licensing.
- **Delivery** charged of preparation, management, stocking, etc..

This internship were conducted at the pole R&I, which is divided into four research programs:

- **Enhanced Media**: content creation including work on digital rendering, animation and 3D.
- **Media Production**: exchange of production including work on digital content search and access.
- **Media Delivery**: broadcasting content and services with innovative and highly secure solutions, including work on home networks, management of content & services and optimizing the user experience.
- **Exploratory**: research on topics that have no directly application but will become a service that Technicolor can offer its customers.

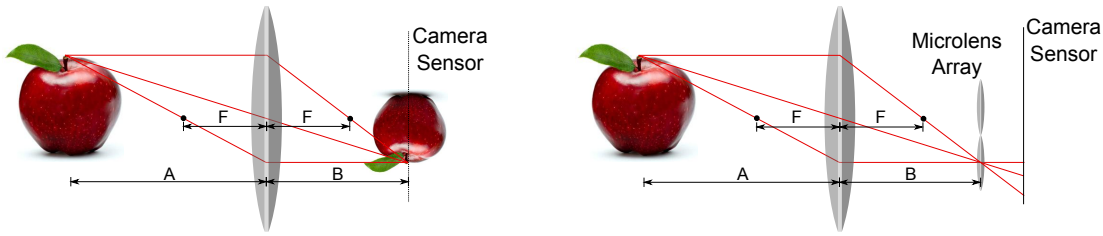
The work we developed falls into the category: Exploratory.


technicolor
R&D Rennes
Technology
Delivery
Research & Innovation (R&I)
Intellectual Property & Licensing (IPL)
Enhanced Media
Media Production
Media Delivery
Exploratory
Patent Operations
Infringement Analysis
Set Top Box
Digital Home
Engineering & Planning
Support & Planning
Support Functions
IPTV

4 The light field acquisition on the Lytro camera

4.1 Light Field parametrization

In a traditional camera, the main lens maps the 3D world of the scene outside of the camera into a 3D world inside of the camera. This mapping is governed by the well-known lens equation:



(a) Traditional camera

(b) Plenoptic camera

Figure 3: Imaging of the scene

$$\frac{1}{A} + \frac{1}{B} = \frac{1}{F} \tag{4.1}$$

Where A and B are the distance from the lens to the object plane and to the object image plane, respectively, and F is the focal distance of the lens. This formula is normally used to describe the effect of a single image mapping between two fixed planes. In reality, however, it describes an infinite number of mappings. That is, every plane in the outside scene is mapped by the lens to a corresponding plane inside of the camera. When a sensor is placed inside the camera at the image plane it captures an image where all the points in the scene at a distance A to the lens are in focus at B.

To accomplish the task of capturing the light field, the Lytro camera have an array of microlenses at the image plane. Each microlens in turn focuses on the main lens, which from the perspective of a microlens is at optical infinity. Rays focused by the main lens are separated by the microlenses, projecting a tiny blurred image on the sensor. At their point of intersection, rays have the same position but different slopes, depending from where they came from. This difference in slopes causes the separation of the rays when they pass through a microlens-space system according to their incident angle. Then this new positional information is captured by the sensor. The position swap at the sensor represents the angular information.

In this fashion, the plenoptic function of the scene is sampled by the microlens array. The spatial sampling is determined by the displacement of the microlenses and the angular by the photosensitive cells at the sensor. This four dimension sample of the plenoptic function is called light field.

For the sake of clarity, the light field can be represented using the two planes representation [15], where each light ray is parametrized by the intersection points in each of the planes, shown in Figure 4.2.

The spatial plane is considered to be the microlenses array plane, and the angular plane is placed at the level of the main lens. Therefore the vector $\overrightarrow{p}$ describes the spatial position of the ray and the

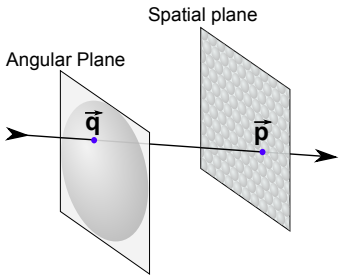


Figure 4: Geometric representation of a light ray using two planes.

vector $\overrightarrow{q}$ the position from where it passed through the main lens. This representation can be thought as a collection of images with different perspectives. If we fix the value of $\overrightarrow{q}$, all points on the spatial plane ($\overrightarrow{p}$) are looking at the same portion of the main lens and are called a view (see Figure 4.3).

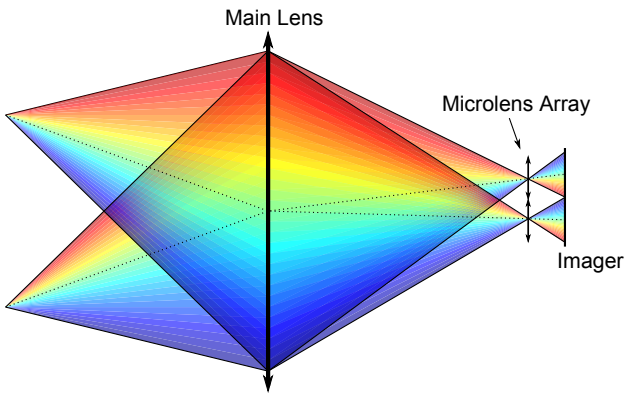


Figure 5: The plenoptic cameras are capable of capturing several views of a scene at the same time. The spatial dimension of the images are sampled at the microlens array plane (spatial plane). The light crossing the up side of the main lens (red rays), for example, fall on the bottom of the subimage formed behind each microlens. Gathering only the information from this portion for each microlens we can reconstruct the image of the scene as view from the perspective of the up side of the main lens.

4.2 Image formation and refocusing

Let $r(\overrightarrow{p}, \overrightarrow{q})$ be the light field function captured by the camera, providing the radiance of a ray crossing points $\overrightarrow{p}$ and $\overrightarrow{q}$. We can perform some simple processing at this function. Its rich information could be used, for example, to calculate how the image would look if captured by a traditional camera with a sensor placed at the microlenses array plane.

The effect of the photosensitive cell is to integrate the light rays coming from all directions. Let B be the distance between the main lens and the microlenses plane, then the equivalent image captured by a traditional camera can be calculated as in [29, 34]:

$$I(\overrightarrow{p}) = \frac{1}{B^2} \iint r(\overrightarrow{p}, \overrightarrow{q}) P(\overrightarrow{q}) cos^4(\theta) d\overrightarrow{q} \tag{4.2}$$

where $P(\overrightarrow{q})$ is an aperture function and θ is the angle of incidence of the ray with the spacial plane.

Using the assumption that the incident rays have small incident angles we can approximate $cos^4(\theta)$ by 1, and further simplify the equations by ignoring the constant $1/B^2$, to define the image formation as:

$$I(\overrightarrow{p}) = \iint r(\overrightarrow{p}, \overrightarrow{q}) P(\overrightarrow{q}) d\overrightarrow{q} \tag{4.3}$$

We can go further in our analysis and try to determine how the light field function appears when seen at another image plane (Figure 4.4). With the assumption that the light rays are not blocked while travelling through space inside the camera, we can determine the synthetic light field radiance $r'(\overrightarrow{p}, \overrightarrow{q})$ at a plane αB distant from the original plane. From triangle similarity, we have:

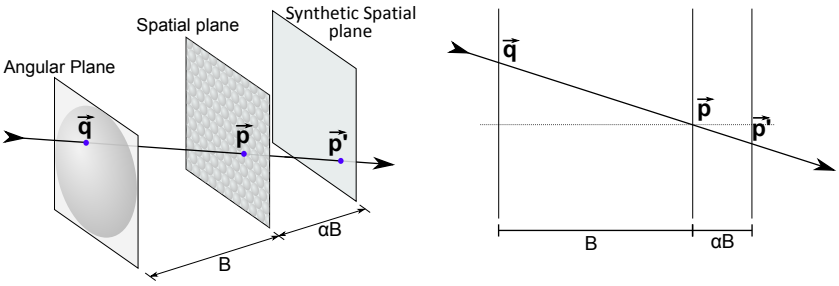


Figure 6: Propagation of the light field (represented in two dimensions for simplification)

$$\begin{aligned} \frac{\overrightarrow{p} - \overrightarrow{q}}{B} &= \frac{\overrightarrow{p'} - \overrightarrow{p}}{\alpha B} \\ \therefore \overrightarrow{p'} &= (1+\alpha)\overrightarrow{p} - \alpha\overrightarrow{q} \end{aligned} \tag{4.4}$$

On this synthetic light field the position from where the light is crossing the main less remains the same, therefore $\overrightarrow{q}$ is unchanged. The value of $\overrightarrow{p}$ can be found by equation (4.4), resulting in:

$$r'(\overrightarrow{p}, \overrightarrow{q}) = r\left(\frac{\overrightarrow{p} + \alpha\overrightarrow{q}}{1+\alpha}, \overrightarrow{q}\right) \tag{4.5}$$

Replacing r in Equation (4.3) by (4.5) we can determine how a traditional camera would perceive the image if focused at the synthetic new image plane of r':

$$I(\overrightarrow{p}) = \iint r\left(\frac{\overrightarrow{p} + \alpha\overrightarrow{q}}{1+\alpha}, \overrightarrow{q}\right) P(\overrightarrow{q}) d\overrightarrow{q} \tag{4.6}$$

Equation (4.6) reveals one of the advantages of using plenoptic cameras: The possibility of refocusing the photos after the shot. At the same time, adjusting the aperture function $P(\overrightarrow{q})$ can be used to play with the depth of field at the synthetic image $I(\overrightarrow{p})$.

Equation (4.6) hides one principal characteristic of the plenoptic cameras: The trade-off between spacial and angular sampling [13]. Since both angular and spatial information are sampled at the same CCD sensor, the more angular information we have the less spatial information we can capture.

The best way to observe how focusing can help to create higher resolution images is analysing the epipolar plane image [8, 22, 32, 37], described next.

Figure 7: Photo refocusing by Lytro.
Credit Lytro Gallery: `https://pictures.lytro.com/lytroweb/pictures/431135`

4.3 The epipolar plane image

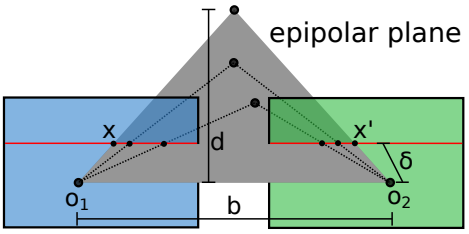


Figure 8: The two cameras are indicated by their centers (O_1 and O_2) and image planes. A point in the three-space forms with the centers of the cameras an epipolar plane. All points falling in this plane will be imaged at the coresponding epipolar line (in red) and the difference of emplacement at one image plane to another is a function of the depth of the object in the scene.

Two views of the same scene are in *epipolar geometry* if all the epipolar lines are parallel. Considering Figure 4.6, if an object intersects the first image plane at x, its intersection with the second image plane will occur at:

$$x' = x - \frac{b\delta}{d} \tag{4.7}$$

In our camera we have available a sequence of image planes whose center are linearly distant from each other. Motivated by this approach we can extract the epipolar lines of each view and organise them into a plane, the epipolar plane image, as in Figure 4.7. It is important to observe that the views are also distributed on the vertical, been possible to generate an epipolar plane image volume [8]. This volume is equivalent to a simple 3D slice through the general 4D light field of a scene [42]. But for simplicity of visualisation we are going to consider just the 2D representation, with no loss of generality.

These layers are a powerful way to describe the parallax of objects in scenes. They capture local coherence, and also make occlusion events explicit. We can extend Equation (4.7) for an array of cameras linearly disposed on the space. Lets assume that an object in the scene is mapped to the position x_0 on the first camera and that b is the distance between two adjacent cameras. Therefore, the corresponding

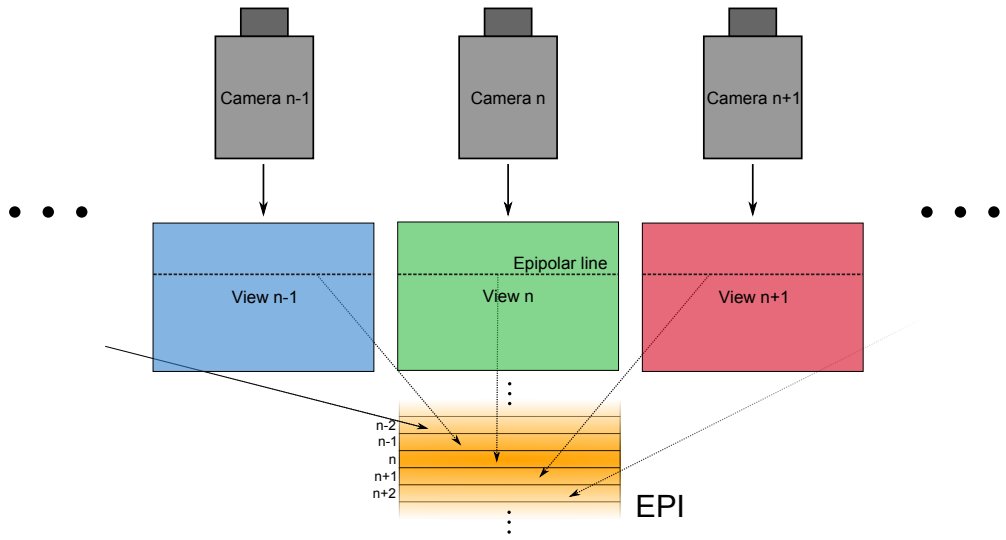


Figure 9: The extraction of the epipolar plane image (EPI) from a sequence of image planes: The center of each camera are displaced horizontally on the space, with a constant distance from each other. Selecting all corresponding epipolar line from each image plan and arranging them as schematised by the figure we obtain the epipolar plane image

position on the image plane for the n-th camera is:

$$x_n = x_o - n\frac{b\delta}{d} \tag{4.8}$$

Equation (4.8) shows that on these framework there is a linear decency between the position where an object from the scene is mapped on the image plane and the camera capturing this information. This create linear structures on the epipolar plane image, which inclination is related to the depth of the object on the scene:

$$\frac{dx_n}{dn} = -\frac{b\delta}{d} \tag{4.9}$$

The EPIs were developed based on the classical multiview stereo but it can be used for the cases of plenoptic cameras as well since they are capable of providing a sequence of views equally distributed on the horizontal and vertical the same way as if several cameras were employed. The main difference between the two approaches are on the relationship between depth and disparity due to the view formation on the plenoptic camera. If the scene, for example, falls at the focal plane of the camera, the objects on the scene will focus on the microlens array and the subimages formed behind each microlens will be composed of the light rays comming from the same point on the scene. Since the pixels of the subimages belong to different views and the position of the pixel on the views is equivalente to the position of the microlens on the array this will result in a null disparity. For objects falling before and after the focal plane we are going to obtain positive and negative disparities. But as the distance between the views is

still constant we obtain a similar result of that shown by Equation (4.9):

$$\frac{dx_n}{dn} \propto \frac{d_o - d}{d} \tag{4.10}$$

where d_o is the depth of the focal plane.

This imply that on the plenoptic cameras, the lines formed on the EPI will have either positive, negative and null slope, depending on the depth of the objects on the scene. Figure 4.8 represents a synthetic EPI from a scene with three objects captured by a plenoptic camera.

In computer vision, the EPI were first proposed as a method for video compression, where each layer is separately coded and predicted using an affine motion model [43]. A more geometric interpretation was introduced by Baker et al. [2], who combined the idea of layers with a local *plane-plus-parallax* representation.

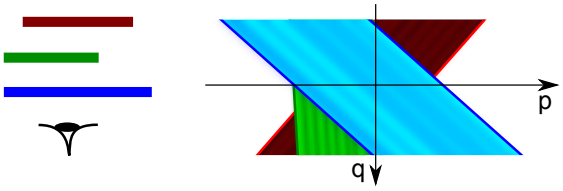


Figure 10: Synthetic epipolar plane image (EPI) of a scene with three objects. The inclination of the lines are dependent of the depth of the objects on the scene. Changing the value of q is equivalent to change the observer position.

We can interpret Equation (4.6) with this new tool. When objects fall at the focal plane, they generate vertical lines on the EPI. Integration along the $\overrightarrow{q}$ dimension will result in a sharp image in this case. Otherwise, if the object does not fall at the focal plane, the contribution from other views will not fall at the same position (inclinated lines on the EPI), resulting in a blurred image. Equation (4.5) act rearranging the inclination of these lines in order to obtain sharp images.

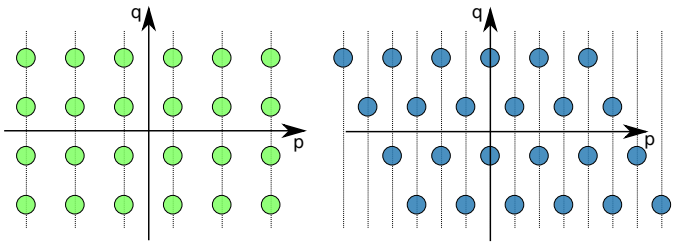


Figure 11: The image formation from the sampled light field. Each dot represent a pixel captured by the plenoptic camera that are distributed through the angular and spatial position. To compute the image as seem by a traditional camera we integrate the light field using Equation (4.6). On the figure, the integration is equivalent to sum the contribution of the samples on the vertical lines. We can observe that the spatial sampling of the refocused image on the right is twice the frequency of the image on the left

A gain in spatial resolution can also be acquired by refocusing the light field. Lets take for example Figure 4.9, where a discrete light field, as obtained by Lytro camera, is shown. If the light field is already in focus, the spatial sampling we will obtain is that determined by the displacement of the microlenses on

the array. On the other hand, refocusing the light field as in Equation (4.5) we can acquire higher spatial sampling, the theoretical maximum been the multiplication of the spatial and angular sampling of the light field.

An all in focus image could be acquired using in Equation (4.5) an α value as a function of the position $\overrightarrow{p}$, creating a synthetic deformed focal plane where all objects are in focus. According to the value of $\overrightarrow{q}$ used to regularise the light field the image can be synthesised with different perspectives. Although we observe that the values of the α to be used are in this case a function of the depth of the scene.

As the light field is represented by a collection of views with lower resolution we can use that information to estimate the depth of the scene and use this data to execute a super resolution algorithm in order to obtain better quality views.

This work concentrate on the aspects of depth estimation. The super resolution is subject of another project following this one to be executed at Technicolor.

5 Overview of Lytro camera data

5.1 LFP Files

The data provided by the Lytro camera is stored on the LFP file format. There are two types of files: those generated directly by the device, named as ".lfp" and those resulting of a post-treatment by a computer receiving the extension "-stk.lfp".

Files of the first type contains the RAW image captured by the camera, as well as the Meta Data that provides information on the conditions of the picture capture (zoom, temperature, etc..), data for deconding the RAW picture into a RGB format (size,number of bits, color pattern, etc..) and device signature (serial number). The second type contains a set of JPEG images focused on a variety of scene depth, which allow the user to play with the focus and perceptive after capturing the photo. On this work we focus only on the first type of file since it contains the RAW data we employ on depth estimation.

Examples of Lytro LFP files can be found on the Light Field Forum [1] available for download.

To access these data we use the tool lfpsplitter [33], Developed by Nirav Patel in C, which extracted the information stored on the two types of LFP files available and convert them to the format RAW, JSON, JPEG or TXT, according to the data.

The raw picture captured by our camera is a square image of size 3280 pixels by 3280 pixels, coded with 12 bits. A BGGR Bayer color filter is used [3], as in most standard cameras (see figure 5.1).

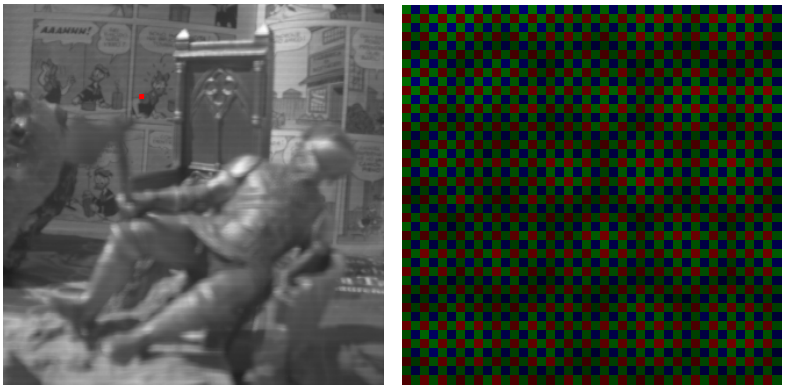


Figure 12: On the left the raw image captured by Lytro camera, on the right a zoom on the red square region showing the images formed under the microlenses array and the Bayer pattern

5.2 Avoiding dead pixels

In practice, we have observed that the raw image contains some dead pixels, that offer a high value of intensity even when not illuminated. Their position were detected by thresholding a photo of a black scene, with the objective of the camera completely closed to avoid light entering it.

[1] http://lightfield-forum.com/2012/08/lytro-sample-lfp-files-for-download/

They appear on every photo taken by the camera, as we can observe on Figure 5.2, where a sequence of 19 photos were shot and the aberrant pixels detected (A pixel is considered aberrant when the difference of its intensity in relationship with its 8 neighbours is above a fixed threshold). Those positions are equal to the positions detected with the photo taken on the black (with the objective closed). Since they don't acquire any valid information, their value is ignored.

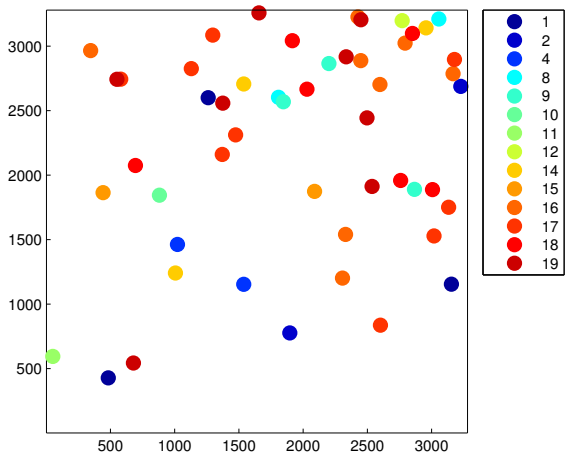


Figure 13: The points represent the position of aberrant pixels on a sequence of 19 photos taken at different scenes. The color represent how many times those pixel positions where detected as aberrant (when its intensity is too different from surrounding pixels). We can observe that the position of this pixels is systematic, appearing on every photo. As this pixels don't carry valid information they are ignored

5.3 Decoding the RAW data to a RGB image

The information needed for the conversion into a color image is provided by the Meta Data (see Appendix B).

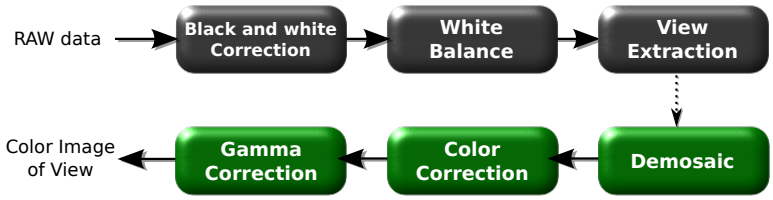


Figure 14: Diagram of the conversion of the raw data to a color image

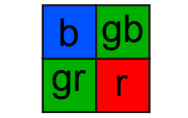
The raw data is a stream of bits resulting from the sweep line by line of the pixels from the photosensitive cells, of size $H * W$, where H is its height and W its width. This image is viewed as a matrix with the same size of the image (H lines, W columns).

5.3.1 Black and white correction

Since each pixel is coded by 12 bits, a range of values between 0 and 4095 is expected, but the sensor is not able to cover all the range, implying that the black and white pixels are not exactly encoded by the 0

and 4095, respectively. The correction of black and white consists to adjust the value in such a way that the black color is encoded by the value 0.0 and white by the value 1.0.

The Meta Data provides values corresponding to white and black pixels for each pixels on the Bayer filter pattern (b, gb, gr and r) and, therefore, the correction is done separately for these pixels. We denote p_1^i the pixel value after black and white correction, been i one of the four colour channel (b, gb, gr and r), p_0^i the pixel value before correction and v_{black}^i and v_{white}^i the value of the level of black and white color for each channel given by "image → rawDetails → pixelFormat" on the Meta Data.



Bayer Pattern

$$p_1^i = \frac{p_0^i - v_{black}^i}{v_{white}^i - v_{black}^i} \tag{5.1}$$

5.3.2 White balance

To adapt to the illumination of a scene a white balance is then applied to avoid the dominance of a particular color in the final picture. The gain applied to each color is provided automatically by the camera depending on the scene. We consider α^i the gain to be applied (image → color → whiteBalanceGain) and p_2^i the pixel after correction.

$$p_2^i = p_1^i . \alpha^i \tag{5.2}$$

5.3.3 Demosaicing

Until this point we have an image with pixels containing information from a single color channel due to the Bayer pattern. The next steps would be to demosaic this image to obtain an RGB image. The demosaicing consist basically to interpolate the missing information using the color from neighboring pixels. But on the Lytro camera the pixels are illuminated by the light ray deviated by the microlenses array as a function of the incident angle and, therefore, neighboring pixels gather information from different portions of the main lens and correspond to different views. Executing the demosaicing at this point would imply a cross talk between views, generating image artifacts that would spoil the depth estimation later on.

This step is them delayed and can only be executed after the extraction of the views (see section 6). There are also two more steps to be execute after demosaicing in order to obtain the final result.

5.3.4 Color space correction

The sensor sensibility to the spectrum of light changes from one device to another. Therefore the color correction consists to convert the color space obtained by the camera to the standard color space. Let's define $c_0 = [r, g, b]^T$ the pixel color after demosaicing on the color space of the device and c_1 the respective pixel color after color correction, then:

$$c_1 = Mc_0 \tag{5.3}$$

where M is a 3x3 matrix given at (image → color → ccmRgbToSrgbArray) on the MetaData.

5.3.5 Gamma correction

Finally the gamma corrections adapts the contrast of the image, resulting on the final pixel color c_2:

$$c_2 = c_1^{\gamma} \tag{5.4}$$

γ given by (image $\rightarrow$ color $\rightarrow$ gamma).

6 Localisation of the subimages

This section covers the conversion of the image captured at the sensor of the Lytro camera to a set of image views. This process, called demultiplexing, consist to, given a pixel, extract its angular and spatial equivalent position of the light field captured on the plane of microlenses and rearrange the image under the form of views combining pixels capturing light from the same portion of the main lens.

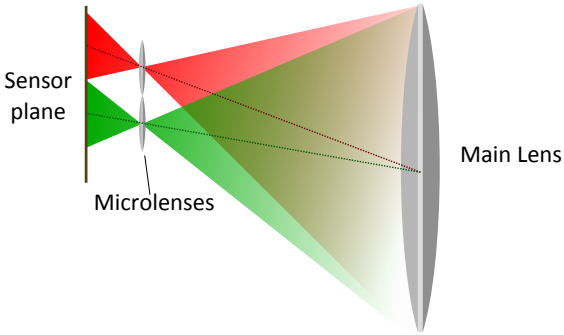


Figure 15: Projection of light passing through the microlenses at the sensor plane forming the subimages

In Figure 6.1 we can observe that the light coming from the main lens, passing trough a microlens, is projected at the sensor forming an image with the same shape of the illuminated portion of the main lens, but scaled. For simplification we analyze the microlenses as pinholes as in Bishop et al. [4]. We do so because scaling and projection are not affected by this assumption.

This projected image of the main lens by each microlens is called subimage, and are distributed the same way as the microlenses array. We can simplify the demultiplexing process by considering the positions of this subimages instead of the microlenses position, what would require as well the knowledge of the distance between the sensor plane, microlenses array plane and main lens.

The center of each subimage corresponds to light rays coming from the center of the main lens and serve as a reference for demultiplexing. The problem becomes then to determine the subimages centers.

To obtain a good spatial resolution a dense distribution on microlenses is required, and an array with more than 120,000 microlenses is present on Lytro camera. As a consequence, a precise model is required.

In the next section we are going to determine a new coordinate system on the image plane that will describe more easily the subimage center's.

6.1 Coordinate system transform

The microlenses of the Lytro camera have a circular shape and are distributed in a way to occupy the maximum possible space to avoid pixels having no information for lack of illumination. This imply that the centers of the microlenses are distributed forming equilateral triangles (see Figure 6.2).

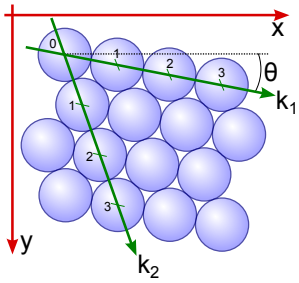


Figure 16: Lens model – The blue circles represent the microlens of the Lytro camera. In red is represented B_1 basis axis and in green the B_2 basis axis.

The subimages have the same distribution of the microlenses as can be seen in Figure 6.3, showing a photo captured by the Lytro camera sensor of a gray (uniform) scene.

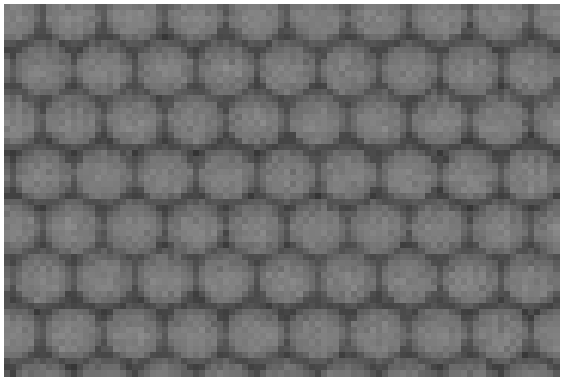


Figure 17: Sample of image captured by Lytro, zoomed to emphasize the subimages distribution

It turns out that the microlenses array can experience a rotation by an angle θ with respect to the image coordinate system due to the building precision. Even θ being small, it can introduce an important error when we take into account the number of microlenses into the camera. This effect is shown exaggerated for clarification in Figure 6.2.

The construction process can interfere as well on the shape of the microlenses making them to be more elongated in one direction than the other, resulting in a ellipsoidal structure.

Let us consider two basis of the Euclidean plane: the canonical or standard basis $B_1 = \{x;y\}$ and $B_2 = \{k_1;k_2\}$. Although B_1 basis is the most natural choice for representing a position it does not adapt very well the geometry of the centers distribution. The B_2 basis overcomes this problem using inclined axis, (k_1,k_2), as represented on Figure 6.2.

We can convert a vector from one basis to another. Let $\overrightarrow{R} = [x,y]^T$ be a vector on the B_1 basis and $\overrightarrow{K} = [k_x,k_y]^T$ its corresponding vector on B_2 basis and $\overrightarrow{C} = [c_x,c_y]^T$ the B_2 origin coordinate system on

the B_1 c.s. Then, the coordinate system transformation equation is:

$$\overrightarrow{R} = T\overrightarrow{K} + \overrightarrow{C} \tag{6.1}$$

where T is the 2x2 transformation matrix.

The matrix T can be decomposed into three simple matrices, $T = BSA$, taking into account the different effects interfering on the geometry of the subimage center distribution: rotation, shape and emplacement, respectively.

6.1.1 The matrix A

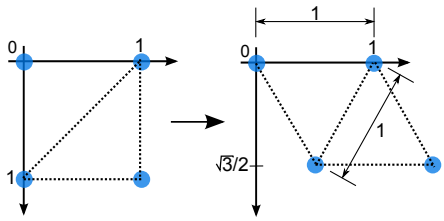


Figure 18: Transformation introduced by matrix A.

The matrix A introduces the emplacement of the centers to the model that, as explained before, correspond to equilateral triangles. Figure 6.4 schematize this transformation imposed by A. The most important effect of this transformation is that a subimage center is placed at the same distance from all the adjacent subimage centers, creating an equilateral triangle emplacement.

As this is a linear transformation, we can determine its terms from a geometrical analysis. From Figure 6.4 we observe that the vector $(1,0)$ is transformed to $(1,0)$ and the vector $(0,1)$ to $(1/2, \sqrt{3}/2)$. Mathematically:

$$A\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix} \;;\; A\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1/2\\ \sqrt{3}/2\end{bmatrix}$$

$$\therefore A = \begin{bmatrix}1 & 1/2\\ 0 & \sqrt{3}/2\end{bmatrix} \tag{6.2}$$

6.1.2 The matrix S

The matrix S determines the distance between adjacent subimages to the model and interfere on the eccentricity of the distribution of centers. Let d_h and d_v be the horizontal and vertical shears:

$$S = \begin{bmatrix}d_h & 0\\ 0 & d_v\end{bmatrix} \tag{6.3}$$

6.1.3 The matrix B

Finally, the rotation by an angle θ of the array is introduced by matrix B:

$$B = \begin{bmatrix}cos(\theta) & -sin(\theta)\\ sin(\theta) & cos(\theta)\end{bmatrix} \tag{6.4}$$

The transformation in Equation (6.1) is determined then by d_h, d_v, θ, and $\overrightarrow{C} = [cx, cy]^T$, called the model parameters.

6.2 Determining the model parameters

The parameters are estimated from an raw image captured by the Lytro camera after applying the black and white correction and the white balance, taken from a gray scene. This results in images similar to Figure 6.3. Macroscopically, the raw data is essentially the same as a conventional photograph. Microscopically, however, one can see the subimages of the main lens aperture captured by each microlens.

To estimate the parameters we use two different methods, which differs from complexity and time of execution. The first one is a global method which tries to estimate the parameters using all subimages at the same time and is more suitable when we don't have an good initial estimation of the parameters. The local method, on the other way, estimates individually the position of the subimages and from this values calculate the parameters, but is dependent of a first initial guess not far away from the real ones.

Both methods take advantage of the fact that the pixels capturing light rays coming from the board of the main lens, or away of its surface, are less illuminated than its neighborhood, as we can see on Figure 6.3. The pixels on the center of the subimages are the most illuminated. This way, the recherche for the subimage centers becomes equivalent to find the positions nearby well illuminated pixels and for which the surrounding pixels have intensity decaying as a function of the distance to this position.

6.2.1 The global method

This method consist in comparing the raw image with a mask which depends on the parameters of the model being estimated. This mask has higher amplitude at the positions where its parameters predict a center of subimage and decay on the edges. The idea is to create a mask that would look like the raw image if the parameters are correct and the solution is found when we obtain a match with the captured image.

Let us call $I_b(\overrightarrow{p})$ the raw image captured by our device, being $\overrightarrow{p}$ a position on the 2D plan, $f(\overrightarrow{p})$ a shape function and $m_{d_h,d_v,\theta,\overrightarrow{C}}(\overrightarrow{p})$ a mask of the same size of the image $I_b(\overrightarrow{p})$ formed by the sum of $f(\overrightarrow{p})$ translated to the predicted subimages position, $\overrightarrow{c}_i$, calculated from the model parameters $\{d_h, d_v, \theta, \overrightarrow{C}\}$:

The shape function tries to imitate the subimage characteristics: high intensity on the center and decay as a function of the distance to the center. The mask is mathematically represented as a convolution with the Dirac distribution centered at the desired positions:

$$m_{d_h,d_v,\theta,\overrightarrow{C}}(\overrightarrow{p}) = f(\overrightarrow{p}) * \sum_i \delta(\overrightarrow{p} - \overrightarrow{c}_i) \tag{6.5}$$

Where $\delta(\overrightarrow{p})$ is the Dirac distribution and "$*$" represent the convolution between two functions.

To verify if a good match is obtained we use the quality function $\Upsilon(d_h, d_v, \theta, \overrightarrow{C})$. A good match would imply that the peaks of the mask coincide with the centers of the subimages and the valleys to the edges. Therefore the best parameters are obtained when Υ is maximal and can be calculated as in Equation (6.7)

$$\Upsilon(d_h, d_v, \theta, \overrightarrow{C}) = \sum_{\overrightarrow{p}} I_b(\overrightarrow{p}).m_{d_h,d_v,\theta,\overrightarrow{C}}(\overrightarrow{p}) \tag{6.6}$$

$$\{d_h, d_v, \theta, \overrightarrow{C}\} = \underset{d_h,d_v,\theta,\overrightarrow{C}}{\arg\max}\, \Upsilon(d_h, d_v, \theta, \overrightarrow{C}) \tag{6.7}$$

Gaussian : $exp\left(\frac{-\lVert\overrightarrow{p}\rVert^2}{\sqrt{2}\sigma^2}\right)$	
Cosinus power 4 : $\begin{cases} cos^4(\frac{\pi\lVert\overrightarrow{p}\rVert}{5{\cdot}5\sigma}) & \lVert\overrightarrow{p}\rVert \leq 2{\cdot}75\sigma \\ 0 & \text{otherwise} \end{cases}$	
Rectangular : $\begin{cases} 1 & \lVert\overrightarrow{p}\rVert \leq \sigma \\ 0 & \text{otherwise} \end{cases}$	

Table 1: Shape functions

σ	**Gaussian**	**Power 4 Cosinus**	**Rectangular**
5	9503	-480	56879
4	33164	31915	156919
3	67738	78278	127951
2	69164	71653	64804
1	23626	22077	17458

Table 2: Sensibility values for the shape function as a function of σ

.

The shape function $f(\overrightarrow{p})$ plays an important role in the parameters estimation. As the illuminated region on the subimages try to occupy the widest space as possible, using $f(\overrightarrow{p})$ with an strong decay can make it incapable of determining the center precisely. On the other hand, a weak decay wouldn't adapt well to the edges.

We tried three shape functions (see Table 6.1) and we tested them with respect to their sensibility. The sensibility measures how Υ changes when the parameters change. For simplification we adopt as sensibility the variation of Υ when the mask and I_b are matched and when a misalignment of 0.04 pixels on the horizontal is present.

From an image taken at a monochromatic gray scene we estimate the parameters using Equation. (6.6) for all shape functions in order to evaluate their sensibility. The quality function $\Upsilon(d_h, d_v, \theta, \overrightarrow{C})$ is plotted in Figure 6.5.

Figure 6.5 and Table 6.2 show that the sensibility becomes maximal when the σ value offers a good approximation of the subimage shape, which has a radius of approximately 5 pixels. The cosinus power 4 and the Gaussien function have similar behavior. The rectangular function usually have higher sensibility because it applies a constant gain to all pixels on the center. But, since the mask used is discrete, the difficult to represent the strong decay of this function at a precise position makes it less reliable and therefore we preferred not use this shape.

Equation (6.7) is interactively solved with an ascending gradient algorithm that adjust the parameters to fit the mask to the raw image by maximizing the quality function. For our camera, for a photo taken at a gray scene, the estimated parameters are shown on Table 6.3.

The MetaData provides the information about the distance between the microlenses, the size of the pixels and the rotation of the microlenses array estimated by the constructor, shown on Table 6.4. The

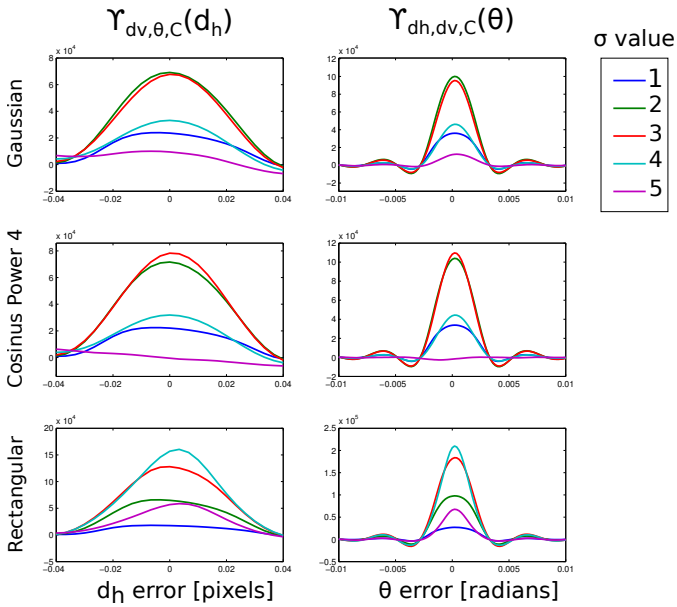


Figure 19: Quality function Υ as a function of the misalignment due to an error at d_h and θ (First and second column respectively) for the three shape function employed (For sake of clarity we subtracted a constant value for every curve, forcing them to be near zero on the extremes of the graphs).

d_h	=	10.00947 pixels
d_v	=	10.01187 pixels
θ	=	0.00141 radians
$\overrightarrow{C}$	=	$\begin{bmatrix} 1639.9129 \\ 1640.4285 \end{bmatrix}$ pixels

Table 3: Estimated model parameters

Pixel Pitch	=	1.3999999761581417e-006 m		
Microlens Pitch	=	1.3999999999999998e-005 m		
Microlens Scale Factor	=	Horizontal	:	1.0000000000000000
		Vertical	:	1.0002111196517944
Rotation	=	0.0013489649863913655 radians		

Table 4: Information extracted from MetaData

rotation we estimated by this method is 4.523% higher than their estimation. The distance between the microlenses is of 10.0000001703 pixels on the horizontal and 10.0021113669 pixels on the vertical, but these values can not be directly compared with ours because our estimation is made from the perspective of the sensor plane. Thus the parameters we obtain correspond to the distance between the subimages and not the microlenses. To compare those values would be necessary to know some other informations, as the distance between the microlens plane and the sensor plane, which is not provided. But the ratio d_v/d_h = 1.00024 is near to the ratio provided by the constructor (1.0002111196517944).

The data provided by the constructor on the MetaData originate from a calibration of the device and are used to refocus the images on their software. We assume that these values can change over time due to wear of the components. Also, changing the zoom and focal plane of the camera modify the subimage formed on the sensor, changing the parameters of the model we developed. Therefore, in order to obtain precise parameters we decided to execute the parameter estimation for every photo taken with our camera using a calibration image, taken from a uniform gray scene after the photo of interest, using the same settings for the camera.

We have observed that the Gaussian and Cosinus Power 4 have a similar behavior, most because their shape is similar. The Rectangular shape function, even though providing high values of sensibility, exhibit some instability for the estimation because of the difficulty to represent its strong decay at a precise position. Therefore, on this work, we choose the Gaussian shape function to execute the parameter estimation.

6.2.2 The local method

The parameters estimation with this method is execute into two steps. First we look through the raw image for the central position of the subimages, individually, by comparing them with a mask with a size limited to the approximative size of the subimage. Then we create a set of central positions that will be next confronted to the model in order to determine the best parameters representing this set of positions.

As the raw image $I_b(\overrightarrow{p})$ employs a Bayer color filter, the pixels from the three color channels can have different intensity. To analyze individually the influence of the pixels disposition on the Bayer pattern on the parameters estimation we devise the raw image into three images, one for each color channel. Hence we define $I_{red}(\overrightarrow{p})$, $I_{green}(\overrightarrow{p})$ and $I_{blue}(\overrightarrow{p})$ as the image obtained from the raw data, containing only information for the pixel belonging to the corresponding color channel. For the pixels where we don't have information for a given channel, its value is set to 0.

The estimation of the center for the subimages is then executed by comparing it with a mask, representing its shape, similarly to the global method, but only taking into account one subimage.

Let $\overrightarrow{c}$ be a candidate to subimage center, $M(\overrightarrow{p})$ the 2D mask used for matching the subimage shape and α_{red}, α_{green} and α_{blue} the weight attributed to each color channel. The quality of choosing $\overrightarrow{c}$ as the subimage center is defined as:

$$Q(\overrightarrow{c}) = \sum_{\overrightarrow{p}} M(\overrightarrow{p}) \begin{bmatrix} \alpha_{red} & \alpha_{green} & \alpha_{blue} \end{bmatrix} \begin{bmatrix} I_{red}(\overrightarrow{p} - \overrightarrow{c}) \\ I_{green}(\overrightarrow{p} - \overrightarrow{c}) \\ I_{blue}(\overrightarrow{p} - \overrightarrow{c}) \end{bmatrix} \tag{6.8}$$

The mask is used to verify the neighborhood of the candidate. Pixels close to the center of the subimage are expected to be well illuminated and the intensity should decay as a function of the distance to the center. So, as the shape function on the global method, the mask weight differently pixels close to the center and pixels close to the subimage edges.

Based on the study we have performed (see Figure 6.5), the mask was choose to have a Gaussian shape, adjusted by its σ value. The set of center positions is initialized with some values well distributed over the image and we adjust them by displacing to the nearest positions maximizing the quality function (local maximum), given by equ. (6.8).

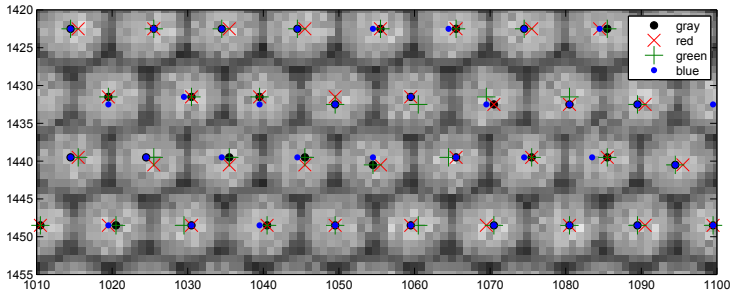
The advantage of this method is that the quality function for this method is a function of the spatial position of the center candidate and not of the model parameters. The size of the mask do not need to

have the same size of the raw image and can have the same size of the subimages. This considerations and the fact that we research just a local maximum makes this method much faster then the global.

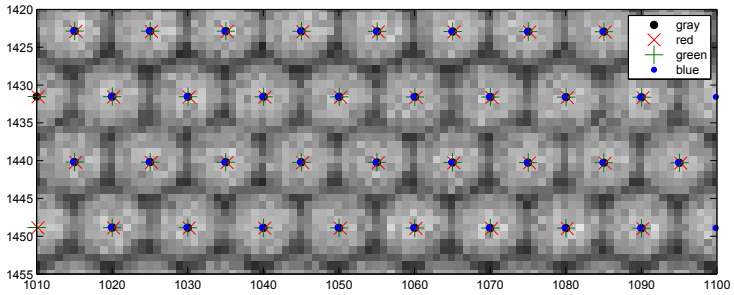
In order to get a well representative set of positions $\vec{c}$, we should initialize their values with positions distributed over the entire raw image. On the same way, the more positions are present on the set the more robust to noise it becomes. The best would be to find the center position of all subimages. In our work we decided to initialize this set with positions calculated from an initial guess of the model parameters.

The values of α_{red}, α_{green} and α_{blue} determine how each color channel will influence the estimation. We can expect that the center position estimated by a single channel is affected by the distribution of the pixels on the color pattern. If the intensity of the pixel red, green and blue are different, using $\alpha_{red} = \alpha_{green} = \alpha_{blue} = 1$ will result on estimations with a preponderance of the effect of the channel with the highest amplitude. Therefore we use the α values to regularize the influence of each color channel, applying a weighting in order to get an unbiased estimation.

The estimation of the subimage center positions, for each color channel individually (α of others channel set to zero) is given at Figure 6.6(a). The mask we use to compute the quality functions has a size of 10x10 because we observed that the subimages have approximately a diameter of 10 pixels. Hence, the estimated center always fall on the intersection of four pixels.



(a) Individual estimation of subimage centers



(b) Estimation of subimage centers from model fitted to data

Figure 20: Estimation of the subimage center position using a Gaussian mask with $\sigma = 3$. The figures show the subimages captured with our camera in a gray scene. The points are the estimation of the central position using different values for α. The gray points are the estimated ceters using $\alpha_{red} = \alpha_{green} = \alpha_{blue} = 1$. The red, green and blue points correspond to the estimated centers with $\alpha = 1$ for the respective color channel and $\alpha = 0$ for other color channels.

Most of the centers estimated using the color channels individually fall on the same position, but we

can observe local tendency appears on some regions. However the tendency changes depending on the center position on the image due to the rotation and scaling of the subimages. This local tendency creates a local error that can be latter attenuated taking into account the positions estimated for all subimages.

We observe this effect on Figure 6.6(b), where the set of center positions were used to estimate the model parameter, and from the model parameters we calculate the center positions of the subimages. The difference between the positions estimated from the different colors are small and we can conclude that the choice of the gain α wont influence significantly the result. Therefore, for the following parameters estimations calculated on this repport, we use $\alpha_{red} = \alpha_{green} = \alpha_{blue} = 1$.

The next step after determining the set of subimage center positions is to confront this values to the model in order to estimate the best parameters adapted to it. This is done verifying the relationship of the representation of those vectors on the R and K-space.

We have available the representation on the R-space (The set we just estimated) but we do not have their representation on the K-space. Although we know that if a vector on the R-space represent a center position of a subimage, them its representation on the K-space should have only integers numbers.

Let us call $\overrightarrow{c}_i$ the i-th center position from the set we have just calculated, represented on the R-space, and $\overrightarrow{k}_i$ its representation on the K-space. From an initial estimation of the parameters we can estimate the value of $\overrightarrow{k}_i$ with the assumption that they are integer numbers:

$$\overrightarrow{k}_i = round[T^{-1}(\overrightarrow{c}_i - \overrightarrow{C})] \tag{6.9}$$

where $round[.]$ is a function that gives the nearest integer to its input value and T and $\overrightarrow{C}$ are dependent of the initial estimation of the parameters $\overrightarrow{C}$, θ, d_h and d_v.

To estimate the parameters we should solve Equation. (6.1) for T and $\overrightarrow{C}$ with the condition imposed by Equation (6.9) for all the values of $\overrightarrow{c}_i$ and $\overrightarrow{k}_i$. As the set $\overrightarrow{c}_i$ has an additive estimation noise, one classical way of solving this problem is minimizing the square root error of Equation (6.9), that is, find the set of parameter where the sum of squared distance of the set $\overrightarrow{k}_i$ to the set $round[\overrightarrow{k}_i]$ is minimal.

In order to do so, let us consider the following notation. We regroup T and $\overrightarrow{C}$ into a single matrix T_e, of size 2x3. We rearrange as well $\overrightarrow{c}_i$ and $\overrightarrow{k}_i$ into K_e, a matrix of size 3xN, and R_e a 2xN matrix, where N is the number of positions present on the set:

$$T_e = \begin{bmatrix} T & \overrightarrow{C} \end{bmatrix} \tag{6.10}$$

$$K_e = \begin{bmatrix} \overrightarrow{k}_1 & \overrightarrow{k}_2 & \dots & \overrightarrow{k}_N \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{6.11}$$

$$R_e = \begin{bmatrix} \overrightarrow{c}_1 & \overrightarrow{c}_2 & \dots & \overrightarrow{c}_N \end{bmatrix} \tag{6.12}$$

Then Equation (6.1) is equivalent to Equation. (6.13):

$$R_e = T_e K_e \tag{6.13}$$

Hence,

$$T_e = R_e K_e^T (K_e K_e^T)^{-1} \tag{6.14}$$

The parameters estimation algorithm takes an initial estimation of the parameters and converges to the best parameters set, as schematized by Algorithm 1.

Input Data: $\overrightarrow{c}_i$, *The set of subimage center positions*
Output Data: $\overrightarrow{C}$, θ, d_h *et* d_v, *The model parameters*
.
% Initialize values to the model parameters
$\overrightarrow{C} = \begin{bmatrix} 1640 \\ 1640 \end{bmatrix}$
$\theta = 0$
$d_h = 10$
$d_v = 10$
.
% initialize $\overrightarrow{R_e}$ *from* $\overrightarrow{c}_i$
$Re = \begin{bmatrix} \overrightarrow{c}_1 & \overrightarrow{c}_2 & \dots & \overrightarrow{c}_N \end{bmatrix}$;
.
while *The model parameters have changed* **do**
% Calculate the model
$T = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} d_h & 0 \\ 0 & d_v \end{bmatrix} \begin{bmatrix} 1 & 1/2 \\ 0 & \sqrt{3}/2) \end{bmatrix}$
.
% calculate $\overrightarrow{k}_i$
$\overrightarrow{k}_i = round[T^{-1}(\overrightarrow{c}_i - \overrightarrow{C})]$;
$Ke = \begin{bmatrix} \overrightarrow{k}_1 & \overrightarrow{k}_2 & \dots & \overrightarrow{k}_N \\ 1 & 1 & \dots & 1 \end{bmatrix}$;
.
% calculate T_e
$T_e = \overrightarrow{R_e}\overrightarrow{K_e}^T(\overrightarrow{K_e}\overrightarrow{K_e}^T)^{-1}$;
.
% from T_e, actualize the model parameters
$[T, \overrightarrow{C}] = T_e$;
$BS = TA^{-1}$;
$d_h = \|BS(:,1)\|$;
$d_v = \|BS(:,2)\|$;
$B = BS \begin{bmatrix} d_h & 0 \\ 0 & d_v \end{bmatrix}^{-1}$;
$\theta = \frac{asin[B(1,2)] + asin[B(2,1)]}{2}$;
end

Algorithm 1: Model parameters estimation from a set of positions

In practice, with our Lytro camera, with a photo shot on a gray scene (different than that used on the global method), the estimated parameters are:

d_h	=	10.00861 pixels
d_v	=	10.01028 pixels
θ	=	0.00143 radians
$\overrightarrow{C}$	=	$\begin{bmatrix} 1639.9128 \\ 1640.2458 \end{bmatrix}$ pixels

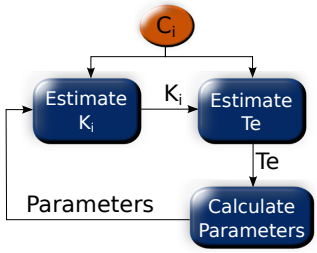


Figure 21: Diagram of the model parameters estimation algorithm from a set of positions

The convergence of this algorithm depends on the initial parameters. Using $d_h = 10$, $d_v = 10$, $\theta = 0$ et $C = \begin{bmatrix} 1640 & 1640 \end{bmatrix}^T$ or parameters from a previous estimation, the algorithm seems to converge correctly. The quality of the result depends on the quality of the set of positions. In this work we use a dense set, trying to cover all subimage positions, resulting in parameters where the additive error is well attenuated. See Figure 6.6(b) for the application of the estimated model parameters from the set represented in Figure 6.6(a).

The two employed method result in similar parameter estimation, but the local method is faster to execute than the global one and is therefore the approach used for all the parameters estimation on the next sections of this work.

6.2.3 Estimated Rotation versus Meta Data

The aim of this section is to compare the estimated rotation with our method and the rotation provided by the constructor. Also, we would like to observe how the parameters change with image zooming and different focal planes.

We took a sequence of 86 photos with our camera, focusing at a printed pattern on an A3 paper, placed at different distances from the device, which was kept at a fixed position. The distance between them was measured for each photo. The same procedure was repeated for three different zoom values (0%, 50% et 100%).

Based on Figure 6.8 we define a rotation reliability margin. We consider acceptable the values of rotation that will imply in an error inferior to 0.5 pixels on the rightmost subimage on the raw data if we consider that the leftmost subimage is correctly centered. As the images have size 3280x3280 pixels, we can assume the distance between the two subimages as been 3280 pixels. Thus, an error inferior to $asin(0.5/3280)$ is allowed.

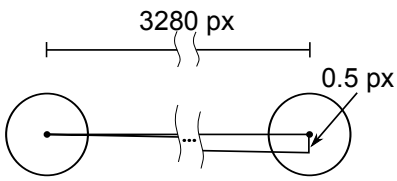


Figure 22: Leftmost and rightmost subimages present on the array and the acceptable rotation error.

Figure 6.9 is a plot of estimated rotation and that provided by the MetaData. The rotation of the microlens array seems to be constant when varying the focal plane and the estimated rotation is always bigger than the Meta Data rotation for our camera, would could be due to an calibration error. Although we observe that the rotation provided at the Meta Data is consistent with our reliability margin.

Contrary to the rotation, the other parameters change with the distance of the camera to the focal plane on the real world and the zoom because the subimages are scaled with the change on the settings of the

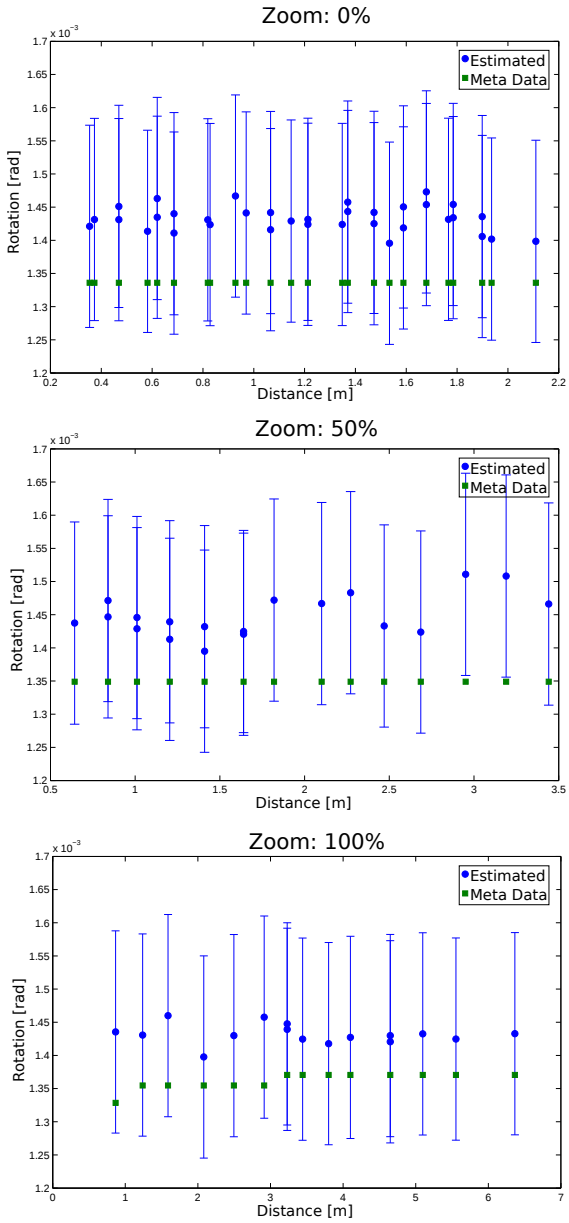
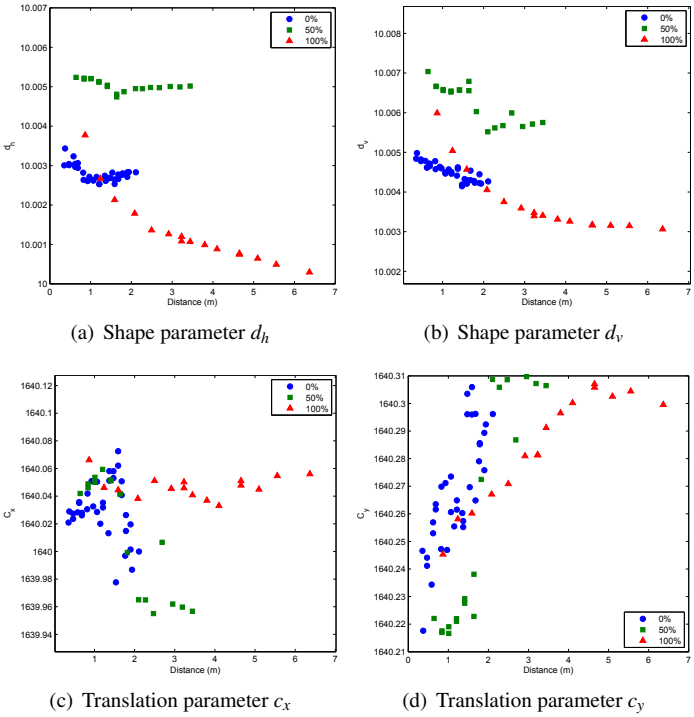


Figure 23: Estimated rotation using the local method versus the rotation provided from the Meta Data for 3 different zooms. From up to down the zoom is 0%, 50% et 100%.

camera to avoid cross talk between views (overlap of the illuminated regions formed by each microlens), as is showed by the curves in Figure 6.10.

The values of d_h and d_v are reduced with the grow of the distance between the camera and the focal plane, but c_x and c_y present a more complex behavior. Predict the parameters would require a deeper analysis of the camera configuration. Therefore we prefer to estimate the model parameter for each photo

taken with our camera in order to obtain precise parameters describing the position of the subimages.



(a) Shape parameter d_h

(b) Shape parameter d_v

(c) Translation parameter c_x

(d) Translation parameter c_y

Figure 24: Evolution of model parameters with the distance of the camera to the focal plane for three values of zoom

7 Demultiplexing the RAW image into views

Once we know the model parameters we can determine the center of each subimage easily. Indeed a vector in the K-space corresponds to the index of a subimage, expressed as an integer number. Using Equation (6.1) we can determine the center of the subimage from its index and rearrange the pixels in order to create the view.

We know that the position on the center of the subimage is receiving light coming from the center of the main lens. Its pixels are mapping the main lens surface at positions linearly dependent on their position on the subimage, as schematized in Figure 7.1.

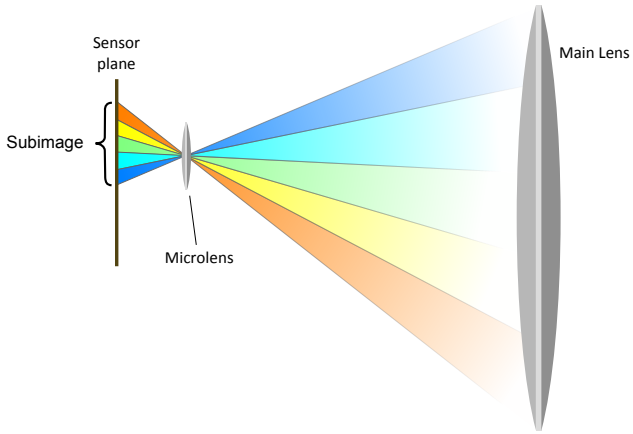


Figure 25: Mapping of the pixel at the subimage on the main lens surface, represented for five pixels, for simplification

The pixel sensitive cell works as an integration device, gathering all the light entering it and representing it by a single information. Thus, each pixel correspond to a square window at the main lens, adding blurring to the views. Using cells of sufficient small size would create an effect similar to a pinhole camera, resulting in all-in-focus views. We will assume that this is true for our camera as an simplification.

The subimages on the Lytro camera have approximately 10 pixels of diameter. Thus we will adopt that all subimages can be represented as in Figure 7.2. As the diameter is a even number this implies that there is no central pixel.

The spatial sampling of the light field is made by the microlenses. To render a view we need to determine the position that the light rays, crossing a given region of the main lens, intersect the microlens array plan. This implies that the emplacement of the pixels on the rendered views will copy the emplacement of the microlens they crossed. Therefore, the pixel belonging to the same view are rearranged according to the position of the subimage from where they come from, as the subimages are arranged similarly to the microlens. The process is represented on Figure 7.3.

Notice that the distribution of the microlens in a equilateral triangle shape adds a complexity to the

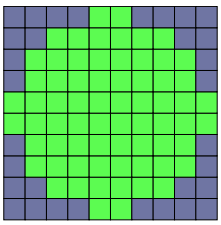


Figure 26: Representation of the subimage in pixels

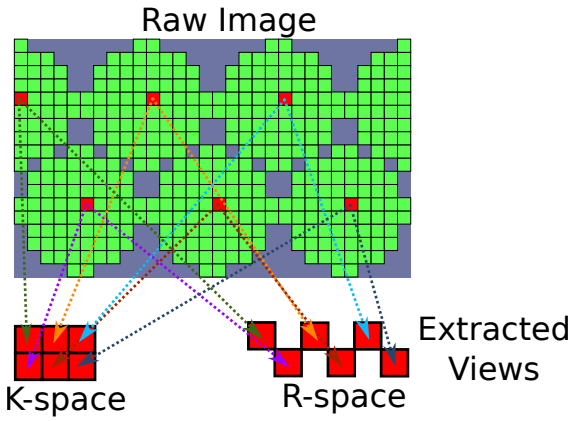


Figure 27: K-space and R-space mapping of the pixels between the subimage and the extracted view

extraction of the views. We want to create views representing well the image they are capturing by distributing the pixels on the image living behind the less as possible regions uncovered by pixel information.

One simple idea is to extract the views on the K-space. Considering this reference system, the constructed view do not have pixels without color information. The main problem of this approach is that it generates inclined images (see Figure 7.5(a)).

On the other hand, if we try to represent the views using the R-space there will be always some missing information as shown in Figure 7.3. To add as less as possible invalid pixels (pixels without information) the solution we found was to use an horizontal sampling $\sqrt{3}$ higher than the vertical sampling.

Figure 7.4 shows an image captured by Lytro camera, after post treatment (demosaicing and refocus) and Figures 7.5(a) and 7.5(b) are the views extracted from the same photo, using the representation on the K and R-spaces, respectively. For the image on the R-space we interpolated the missing points with the mean of the neighboring pixels to help the visualization.

Figure 28: Photo taken from Lytro camera after post processing. The image was generated by the software provided by the constructor, which enables us to create a coloured image focused at the desired focal plane

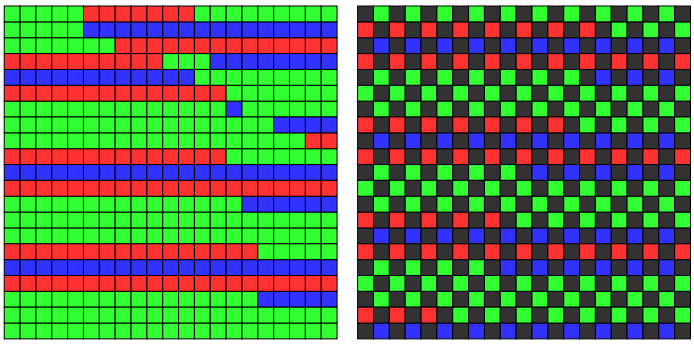
We can see from those figures that there are regions presenting some effects looking like horizontal

(a) View in the K-space (b) View in the R-space

Figure 29: View extracted from the photo used to generate Figure 7.4

lines. As explained on the Section 5.3 we can not execute the demosaicing of the pixels before the view extraction. Therefore the pixels on the views contain information about just one color channel and the lines are the result of the jumps from one color to another.

The resulting color pattern is represented in Figure 7.6. The distance of two subimages on the horizontal is about 10 pixels, that is an even number. This creates on the views the tendency of having long lines of the same color. On the vertical, on the other hand, this distance is about 8.66 pixels, that is near to 9, resulting in a faster changing of the color.



(a) K-space (b) R-space

Figure 30: Resulting color pattern on the views

To reconstruct the full color image a demosaicing is necessary, interpolating the missing information. The risk at this point is to introduce artifacts due to the distance between two pixels of the same color because we intercalate between regions rich in only one color.

Another problem observed during view extraction is related to the assumption that all subimages can be represented as in Figure 7.1, what is not exactly true. The values of center position computed from the model parameters are allowed to have any value in the continuous set of real numbers, but the positions on the subimages are discretized by the pixel positions. To extract the corresponding pixel of a view from the subimage without interpolation, to avoid cross talk, we are obligated to take the pixel nearest to the desired position. This can introduce some artifacts on view rendering, specially for objects of the scene falling far from the focal plane.

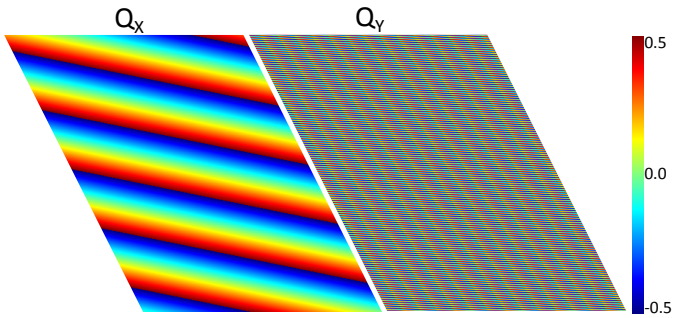


Figure 31: View emplacement error introduced on each direction for one given view, extracted on the K-space

7.1 The vignetting correction

The subimages suffer as well from the vignetting effect, that is a phenomenon of brightness attenuation of light rays as they cross the optical system. The result is a gradual decrease in light intensity towards the subimages periphery.

To correct the vignetting we can apply a gain to the pixels as a function of their position to obtain a more regular image, either by using the values from a LUT (Look Up Table) or by a functional approximation of the correction factors distribution. Most methods for vignetting correction use a parametric vignetting model to simplify estimation and minimize the influence of image noise. Typically used are empirical model such as polynomial functions [38, 16] and hyperbolic cosine functions [45].

To estimate the vignetting on our images we use a calibration photo taken from a scene that try to approach the ideal case of a completely white image. The calibration photo is taken for every scene acquired with our camera since both vignetting and model parameters change with the zoom and focal plane of the camera.

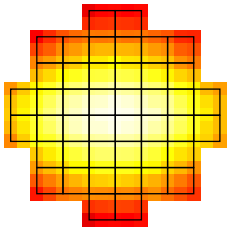


Figure 32: Vignetting effect image

There are two types of vignetting present on the images. The illumination of the subimages may vary with its position on the RAW data, that will introduce a vignetting on the rendered views, and the illumination of the pixels on the subimages decay as we approach to its periphery, what imply in a difference of intensity from one rendered view to another. The second type of vignetting is more intense than the first one.

In this work we do not treat the first type of vignetting because our calibration images present some changes of illumination from the way the calibration photo was taken, what could imply in erroneous estimation of its effect and the introduction of illumination artifacts, but the vignetting on the the subimages can be easily determined. Therefore we use the term vignetting on this work to refer uniquely to the vignetting observed on the subimages.

To eliminate the noise effect on the estimation of the vignetting we compute the mean intensity value of the pixels falling on corresponding positions on each subimages. This is equivalent to compute the average subimage intensity. The resulting image is similar to that shown in Figure 7.8 (we ignore the pixels from the edges of the subimages because they are affected as well by mechanical vignetting due to the camera aperture, that is, to avoid cross talk between subimages the aperture is adjusted and the pixels on the frontier between two subimages have their light rays blocked by the aperture). Normalizing this

image, that is, dividing it by its maximum value, will generates an image of the vignetting effect from where we can determine the gain to apply to correct its effect.

The vignetting for our camera is more intense on the vertical than on the horizontal, what could be due to a deformation of the lens curvature and/or a rotation error on the positioning of the lens set. Furthermore we observed that the shape of the vignetting is influenced by the position of the subimage on the RAW image (see Figure 7.9), what results mostly in a rotation and translation of the vignetting center (position receiving the highest light intensity) resulting from aberration of the lens. The color channel of the pixels also influence the vignetting (see Figure 7.10). The vignetting on the red channel has a center translated to the right, when compared with the green channel, and the blue channel has its center translated to the left.

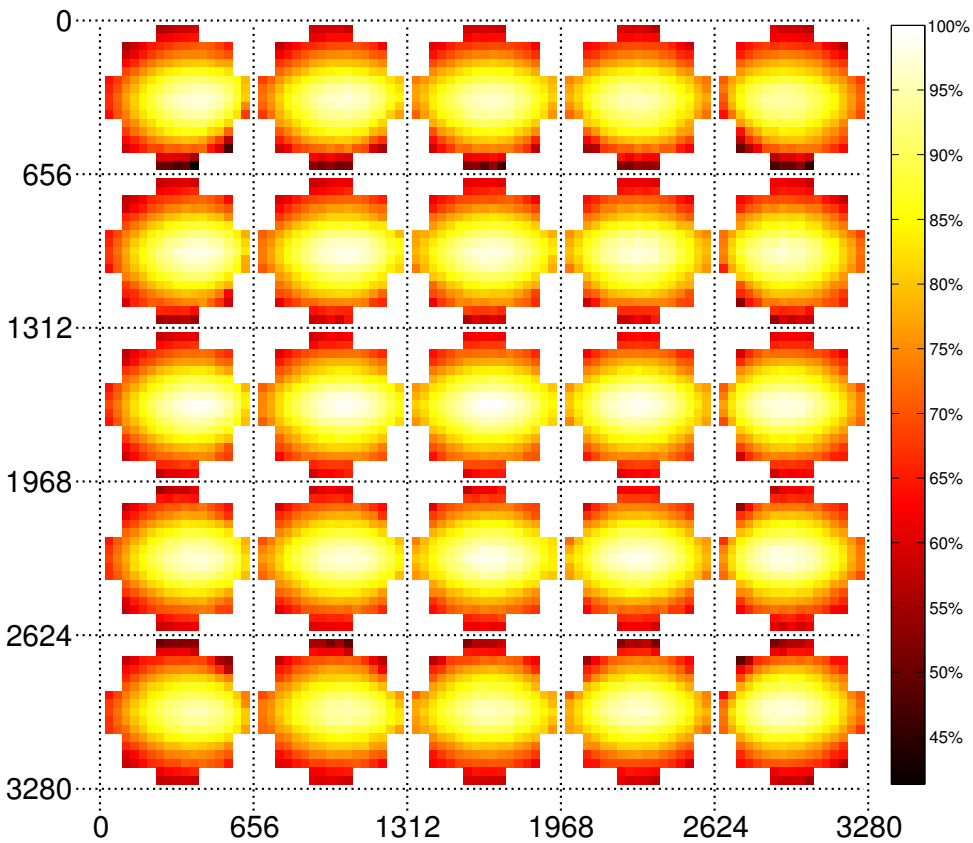


Figure 33: Vignetting effect image (VEI) according to the position of the subimages on the RAW image. The RAW image where dived into squares of size 656x656 and the VEI where calculated by the mean intensity value of the red pixel for the subimages falling on the corresponding region.

To correct the vignetting we compute the vignetting effect image by averaging all subimages intensity for each color separately. We model the vignetting by a hyperbolic cosinus function of the form:

$$cos^4\left(\sqrt{\alpha_x^2(x-\delta_x)^2+\alpha_y^2(y-\delta_y)^2}\right) \tag{7.1}$$

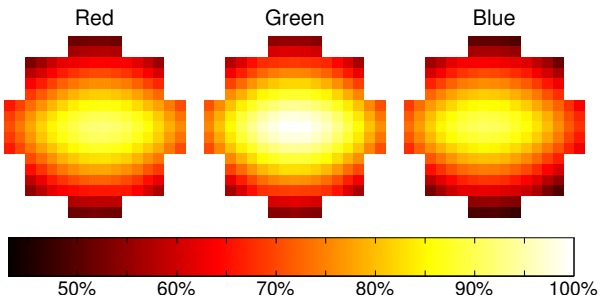


Figure 34: Vignetting effect image for each color channel

where (x, y) is the position of a pixel on the subimage and α_x, α_y, δ_x and δ_y are the vignetting parameters.

During view rendering, we apply a gain that is the inverse of Equation (7.1) to the pixel of one view in an attempt to reduce its effect. Normalized views are necessary to estimate the disparity and the vignetting correction can eliminate some artifacts, as shown on the next section.

8 Disparity estimation

8.1 The demosaicing problem

Until now we have been pointing out that the traditional solution of demosaicing each subimage captured by the camera before rendering the views implies in cross talk between them, hence degrading the views. The phenomenon of cross talk is more intense for high frequency elements of the scene because the neighbouring pixels on the subimages do not correspond to neighbouring pixels on the views and the texture of the objects is degraded by the interpolation applied by the demosaicing algorithm.

Even though, we have observed in the literature that most authors still use this approach [1, 13, 27, 29] and few works focus on the demosaicing aspect. An example is Georgiev [14], who introduced a demosaicing scheme to be applied after merging the views to create a refocused image, as in Equation (4.6), or, if disparity information is available, it is even possible to create coloured high resolution views where all objects on the scene are in focus.

Our scenario is different from that of Georgiev: his objective was to develop an refocusing approach taking into account the demosaicing problem and ours is to estimate the disparity on the views. We can't use refocused views for disparity estimation because it would result in blur on the images and degrades the quality of the final disparity map.

The demosaicing can only be applied after rendering the views, but due to the distance between two pixels of the same color on the resulting pattern this task usually introduce a lot of color artifacts. Therefore we decided to modify our approach to avoid this problem.

Nevertheless we observed that even without executing the demosaicing, it is still possible to estimate the disparity. The raw data is coded using a Bayer color filter. To extract the views, the pixel are regrouped according to their positions. Therefore, adjacent views will have completely different color patterns because they gather information from adjacent pixels on the subimages, which have different colors. Although, as the Bayer pattern repeats itself with a period of two pixels, on the horizontal and vertical, views distant by two (and its multiples) will be composed by the same color pattern.

Due to the Bayer filter and the microlenses position, the rendered views have a pattern presenting long sequence of pixels of the same color on the horizontal (about 50 pixels long). So we can estimate the correspondence between views with horizontal displacement having the same color pattern, i.e., finding points in the left and right images that are a projection of the same scene point (a conjugate pair) by finding the element on the right image which is the most similar to a given element in the left image.

The disparity is estimated by minimizing a cost function which measures the dissimilarity between the views using an area-based algorithm [20, 11, 10] that tries to match a small image window centered at a given pixel [12] with an equivalent image window on the other image, but shifted. Let I_1 and I_2 be two views horizontally distant, d the disparity value, $W(i,j)$ the weight function on the pixel (i,j), Ω the support of W and $S(x,x+d,y)$ a cohesion function, which is equal to 1 when the pixels being compared between the two views gather information of the same color channel and 0 otherwise. Then the cost

function $CF_{x_0,y_0}(d)$, based on the sum of squared differences [19], at the position (x_0, y_0) is:

$$CF_{x_0,y_0}(d) = \frac{\sum_{i,j\in\Omega} [I_1(x_i,y_j) - I_2(x_i+d,y_j)]^2 W(i,j) S(x_i,x_i+d,y_j)}{\sum_{i,j\in\Omega} W(i,j) S(x_i,x_i+d,y_j)} \tag{8.1}$$

$$x_i = x_0 + i$$

$$y_j = y_0 + j$$

The division by the sum of the window's weight multiplied by the cohesion function is necessary to normalize the cost function since the color pattern may not match exactly due to the shift of the window and the number and weight of the comparable pixels change with the disparity.

The cost function presented in Equation (8.1) can adapt to any pair of views, even those having different color patterns. But when the number of comparable pixels between the views is small, the cost function becomes noise, reducing the quality of the disparity maps we can obtain. That is the reason why we only compare views having the same color pattern.

We avoid as well to estimate the disparity on the vertical because the resulting color pattern presents an intense change of colors from one horizontal line to another, leading to high changes on the number of comparable pixels as we change the value of disparity being tested. The noise samples of the cost function we obtain in this case, for the scenes we tested, resulted in maps with lots of artifacts for the vertical disparity.

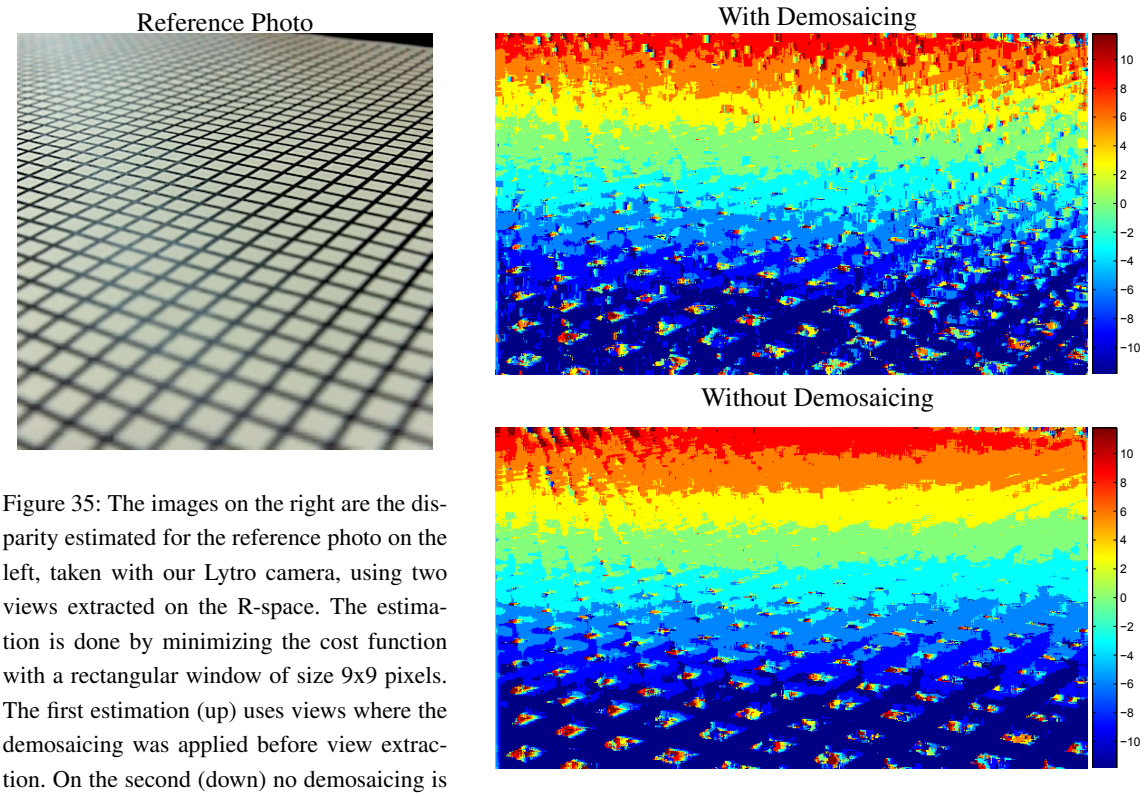


Figure 35: The images on the right are the disparity estimated for the reference photo on the left, taken with our Lytro camera, using two views extracted on the R-space. The estimation is done by minimizing the cost function with a rectangular window of size 9x9 pixels. The first estimation (up) uses views where the demosaicing was applied before view extraction. On the second (down) no demosaicing is applied.

Figure 8.1 shows the artifacts introduced by the demosaicing when executed before view extraction that is an evidence that this is not the optimal approach. The demosaicing algorithm we employed is that provide by MATLAB. We observe as well that both estimations present artifacts due to the low frequency regions of the views since the cost function do not have a well defined minimum position on those regions. The high frequency elements introduce an error to the measure due to the aliasing, as can be seen on the top left of the estimated disparity.

As mentioned in Section 7, the views extracted on the R-space result in half of the pixel without any color information (invalid pixels) (Figure 7.6). Therefore, an odd displacement will cause all valid pixels from one image window to be compared with invalid pixels at the other and vice-versa, resulting in $S(x_i, x_i + d, y_j)$ being zero everywhere and an infinity cost function value. So the cost function is only minimized for even disparity values.

8.2 Multi-View Stereo

Theoretically, from the subimages on the RAW data of the Lytro camera is possible to render 10x10 views of the scene, one for each pixel of the subimage (See Figure 7.2), displaced equally on the horizontal and vertical in perfect epipolar geometry. In this work, the views capturing light from the edges of the main lens, or out of the main lens surface, are ignored because they result in a degraded image. We then obtain a total of 44 views, as represented on Figure 8.2.

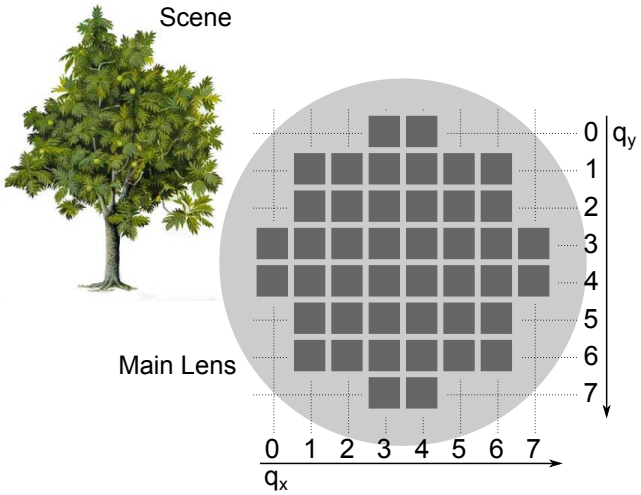


Figure 36: Schematic of the capture of a photo by the camera main lens. Each square represent a portion of the main lens. The image rendered from the light rays crossing a given square correspond to a view. Therefore we use the position of each square to refer to its resulting view (Only the square corresponding to a valid views are represented), noting as $\{q_y, q_x\}$ the view rendered from the light coming from the square at the position (q_y, q_x).

We say that two views are distant by n when their correspondent square region on the lens surface are distant by n regions. For example, views {4,1} and {4,3} are distant by 2 on the horizontal, and views {3,3} and {6,3} are distant by 3 on the vertical.

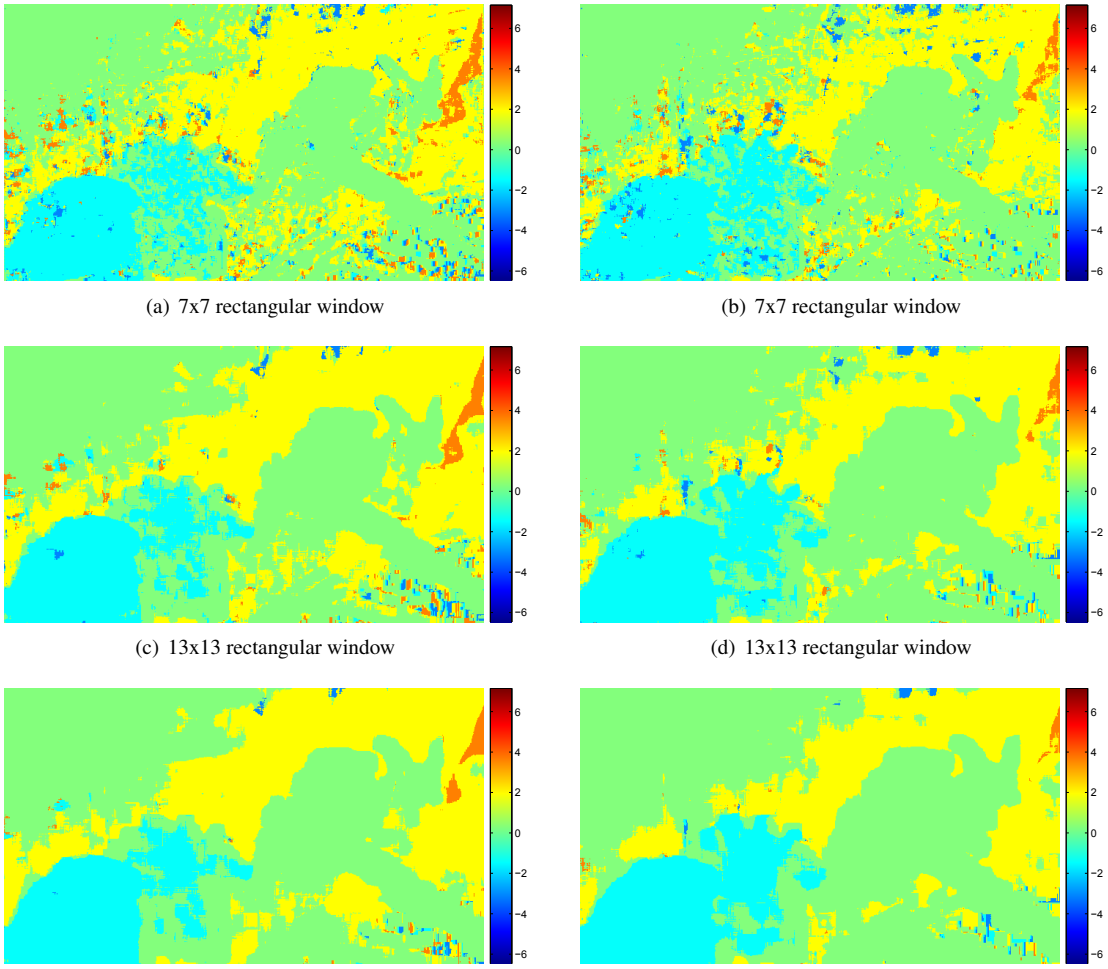
We can compute Equation (8.1) to any pair of views on the same line, but chosen two pair of views distant by an even number is preferred because there will be a higher number of pixels to compare due to

Figure 37: Photo of the scene captured by the Lytro camera. The image was obtained from the raw data after applying a refocusing and demosaicing approach to render the given image. The algorithm used for the task is that provided by the constructor of the camera

the fact that the two views have the same color pattern.

Let us consider, for example, the photo captured from the scene on Figure 8.3. We extracted four views from the raw data on the R-space using the method as described before and we compute the cost function described by Equation (8.1). The results we obtained by minimizing the cost function for views distant by two on the horizontal are shown in Figure 8.4 for different sizes of windows.

We can observe that growing the size of the windows allow us to obtain disparity estimation with less noise, but the resulting disparity loses precision for defining sharp edges between objects (fattening effect).



(a) 7x7 rectangular window

(b) 7x7 rectangular window

(c) 13x13 rectangular window

(d) 13x13 rectangular window

(e) 19x19 rectangular window

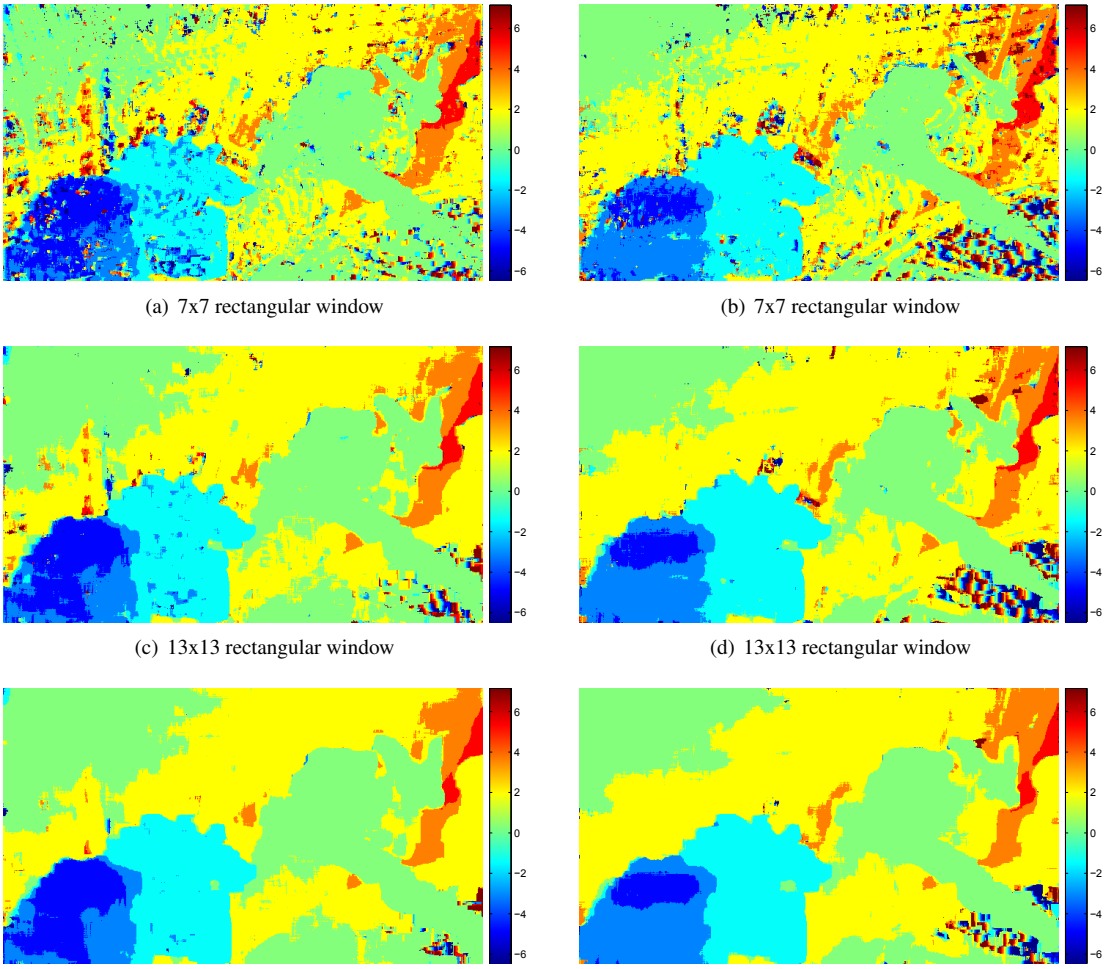
(f) 19x19 rectangular window

Figure 38: Disparity estimation of two views distant by 2 on the horizontal. At the column on the left, the two views employed to compute the cost function are {3,3} and {3,5}, and on the right {3,2} and {3,4}. The disparity value equal to zero correspond to the objects on the scene falling on the focal plane of the camera, and the negative and positive values of disparity correspond, respectively, to objects falling between the focal plan and the camera main lens and between the focal plan and the infinity

The 13x13 rectangular windows seems to accomplish a satisfying trade-off between disparity noise and edge resolution.

Even though the quality of the disparity maps we obtain are low by reason of the views resolution and the color pattern, this is a proof that the disparity estimation is possible without demosaicing the views.

Similar results are obtained when we estimate the disparity with views distant by 4, but in this case, as the distance between the views is twice, the estimated disparity is doubled and has a better precision (Figure 8.5).



(a) 7x7 rectangular window

(b) 7x7 rectangular window

(c) 13x13 rectangular window

(d) 13x13 rectangular window

(e) 19x19 rectangular window

(f) 19x19 rectangular window

Figure 39: Disparity estimation of two views distant by 4 on the horizontal. At the column on the left, the two views employed to compute the cost function are {3,1} and {3,5}, and on the right {3,2} and {3,6}. The disparity value equal to zero correspond to the objects on the scene falling on the focal plane of the camera, and the negative and positive values of disparity correspond, respectively, to objects falling between the focal plan and the camera main lens and between the focal plan and the infinity

Of course using only two views is suboptimal and a better estimation can be obtained if we take profit of the rich information provided by the camera. Let us concentrate on the sequence of views present on a line, since they have only horizontal parallax.

We notice that the obtained disparity map is dependent of the view we chose as reference. For example the disparity from the view {3,3} to {3,5} will have negative values for objects near to the camera and positive for those far away. On the other hand, if we estimate the disparity from {3,5} to {3,3} the behavior will be the opposite. Moreover the disparity between the view pairs {3,3}-{3,5} and {3,2}-{3,4}, choosing the first view as reference, have the same behavior, but will have a slight difference of perspective because of the parallax on the views. As the disparity copies the perspective of the reference

view we will observe parallax on the disparity maps.

Therefore, in order to match all views preserving the same behavior and perspective we chose arbitrarily one view as reference for all pairs of images. The idea is to compare between them views having the same color pattern (Figure 8.6). To preserve the same behavior for all maps we use an unique definition of disparity. In this work the horizontal disparity is the translation we need to apply to a pixel of a given view to find its equivalent position on the view immediately on the right.

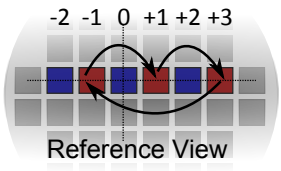
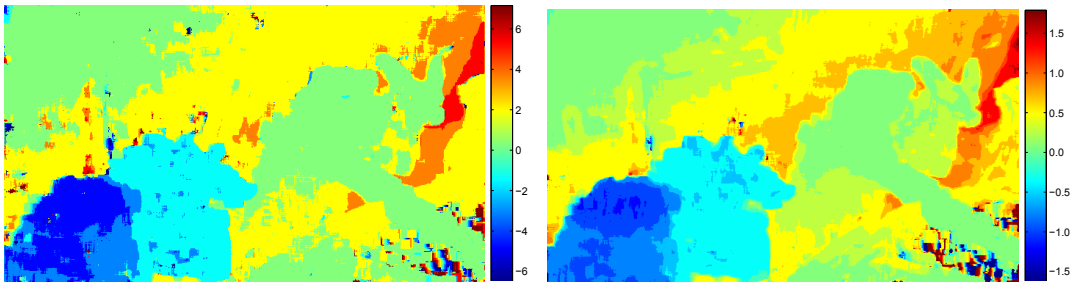


Figure 40: Multi-stereo matching between views on the horizontal. The views in red have the same color pattern and are therefore comparable. The same is true for the views in blue.

Then we modify Equation (8.1) to take into account all the views present in a line. Let Θ be the set of comparable view pairs in a given line and a_n the distance between the view n and the reference view, then:

$$CF_{x_0,y_0}(d) = \sum_{(n,m)\in\Theta} \left(\frac{\sum_{i,j\in\Omega} [I_n(x_i + a_n d, y_j) - I_m(x_i + a_m d, y_j)]^2 W(i,j) S_{(n,m)}(x_i, x_i + d, y_j)}{\sum_{i,j\in\Omega} W(i,j) S_{(n,m)}(x_i, x_i + d, y_j)} \right) \quad (8.2)$$



(a) Pair {3,1}-{3,5}

(b) Line $q_y = 3$; Reference view: {3,3}

Figure 41: Disparity estimation using a rectangular window of size 9x9 pixels: (a) Estimated using two views distant by 4 and Equation (8.1); (b) Estimated disparity using 6 views distributed on the line $q_y = 3$ and Equation (8.2). The pair of views present on the set Θ are: $(\{3,1\}-\{3,3\},\{3,3\}-\{3,5\},\{3,1\}-\{3,5\},\{3,2\}-\{3,4\},\{3,4\}-\{3,6\},\{3,2\}-\{3,6\})$.

We can observe a clear gain on the quality of the disparity carts because six pairs of views are compared at the same time, reducing the noise effect.

8.3 The error map

On the classical stereo vision it is usual to estimate the disparity from the left view to the right view and verify the cohesion with the estimation of the right view to the left. Changing the reference view

modify the cost function, but it is expected to obtain similar estimation on the well textured region of the views and noise estimation for the poor textured regions or those presenting artifacts such as aliasing. Computing the difference between both estimations on the equivalent positions offers a measure of the cohesion between the two maps.

We adopt an similar approach. If we change the views pair on the set used to compute the disparity estimation we modify the cost function. Comparing the difference between the disparity estimation employing different sets of views would provide a cohesion measure similar to the case of stereo vision that we call the error map. We can use this error map to determine how reliable a disparity estimation is. By thresholding this map we can eliminate the noise values, for example, or use this measure for further processing in order to improve the quality of the disparity maps.

To calculate this map we modified the method of computing the cost function. We use an equation similar to Equation (8.2), but we broke the outer sum into individual estimations that will be compared together. This is equivalent to employ a set of views Θ composed by only a pair.

At the low frequency regions of the views the cost function is mostly composed by noise and we can expect that the estimated disparity on those regions will vary from one pair to another. Similarly, on the regions where the frequency of the texture is above the Nyquist limit (half of the sampling frequency) there will be aliasing that introduces minimums to the cost function on incorrect values.

We fixed the set of views Θ to be composed by six pair of views: $\Theta = [\{q_y, 1\} - \{q_y, 3\}; \{q_y, 2\} - \{q_y, 4\}; \{q_y, 3\} - \{q_y, 5\}; \{q_y, 4\} - \{q_y, 6\}; \{q_y, 1\} - \{q_y, 5\}; \{q_y, 2\} - \{q_y, 6\}]$.

To regroup this six disparity estimations d_n on a single value we computed either their mean value (Equation 8.3) or their mean value weighted by the dif measure of each set (Equation 8.4).

The dif measure is the difference of the first and third minimum of the cost function. A high value for this measure correspond to cost functions where the minimum is well defined (high value of second derivative at the minimum) and therefore can provide values with a better confidence. We use the first and third minimum instead of the first and second to avoid the case represented on Figure 8.8, where the actual minimum of the cost function falls between the two first minimums and would imply a value near to 0, even though the minimum of the cost function is well defined but not well estimated by the first minimum.

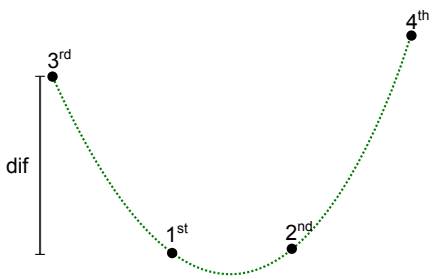


Figure 42: Computation of the dif value

$$d = \frac{\sum_{n=1}^{6} d_n}{6} \tag{8.3}$$

$$d = \frac{\sum_{n=1}^{6} dif_n^2 d_n}{\sum_{n=1}^{6} dif_n^2} \tag{8.4}$$

The error map is computed from the squared difference of all estimations, as defined below:

$$error = \sum_{k=1}^{5} \sum_{n=k+1}^{6} (d_k - d_n)^2 \tag{8.5}$$

The quality of the estimated disparity is also influenced by the algorithm employed to compute the Cost Function. The literature provides a wide variety of algorithms [23, 41, 6, 19]. On this work we decided to restrain to the region based approaches since the views we work on are not demosaicked. Common window-based matching costs include the sum of absolute or squared differences (SAD / SSD), normalized crosscorrelation (NCC), and rank and census transforms [44].

The approaches we chose are: the sum of squared differences (SSD), that is the method we have been currently using that suppose that the images being compared are rectified; zero-mean sum of squared differences (ZSSD) that try to avoid the artefacts produced when the images do not have the same illumination; normalized cross correlation, an algorithm that compare the similarity of the two images and is robust to linear transformations.

These algorithms are implement to take into account the color pattern of the views. The cost function used to estimate the disparity for one pair of views $m-n$ is given below:

- **Sum of Squared Differences (SSD):**

$$CF_{x_0,y_0}(d) = \frac{\sum_{i,j\in\Omega} K\,[I_n(x_i + a_n d, y_j) - I_m(x_i + a_m d, y_j)]^2}{\sum_{i,j\in\Omega} K} \tag{8.6}$$

- **Zero-Mean Sum of Squared Differences (ZSSD):**

$$CF_{x_0,y_0}(d) = \frac{\sum_{i,j\in\Omega} K\left[I_n(x_i + a_n d, y_j) - \tilde{I}_n - I_m(x_i + a_m d, y_j) + \tilde{I}_m\right]^2}{\sum_{i,j\in\Omega} K} \tag{8.7}$$

- **Normalized Cross Correlation (NCC):**

$$CF_{x_0,y_0}(d) = \frac{-\sum_{i,j\in\Omega} K\left[I_n(x_i + a_n d, y_j) - \tilde{I}_n\right]\left[I_m(x_i + a_m d, y_j) - \tilde{I}_m\right]}{\sqrt{\sum_{i,j\in\Omega} K\left[I_n(x_i + a_n d, y_j) - \tilde{I}_n\right]^2}\sqrt{\sum_{i,j\in\Omega} K\left[I_m(x_i + a_m d, y_j) - \tilde{I}_m\right]^2}} \tag{8.8}$$

where $K = W(i,j)S(x_i, x_i + d, y_j)$ and $\tilde{I}$ is the mean value of I on the windows support Ω.

Figure 8.9 shows the shape of the cost function for the algorithms implemented in a particular case and Figure 8.10 the estimated disparity with those algorithms. For the estimation with the SSD we proposed two methods: execute the estimation with and without applying a correction for the vignetting. Very similar results are obtained for the SSD with vignetting correction, the ZSSD and NCC, showing that the vignetting correction was effective to eliminate the artefacts due to the difference in illumination on the views being compared. Therefore we fix the vignetting correction as an standard approach to be applied to the views on all disparity estimation algorithms.

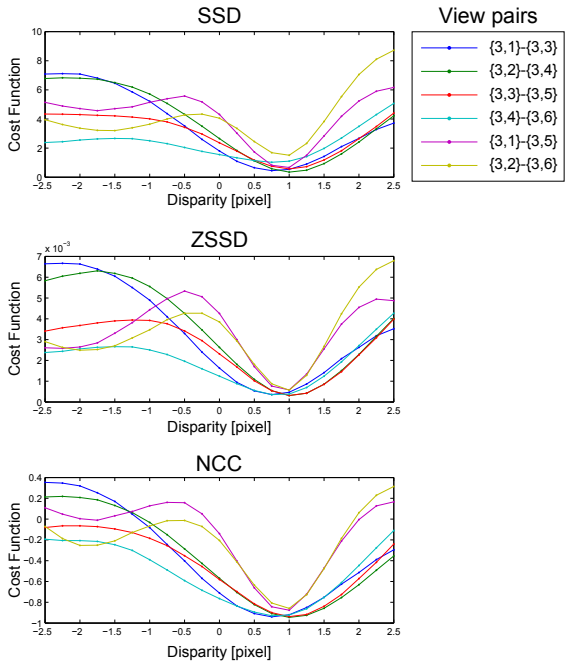


Figure 43: Shape of the cost function obtained at a textured regions of a sequence of views rendered from a photo taken with Lytro camera. The images where resampled using a PCHIP interpolation by an effective resampling value of x8 (x4 in practice). The windows employed is rectangular of size 13x51 ($\approx$ 13x$(4*13)$, chose to produce similar results of using a windows of size 13x13 when the views are not interpolated).

We restrained our analysis to just a qualitative comparative between the different algorithms. A best comparative would be done numerically computing the error of the estimated disparity to a ground truth for each algorithm. We delay this approach to the next section because we need to use their results.

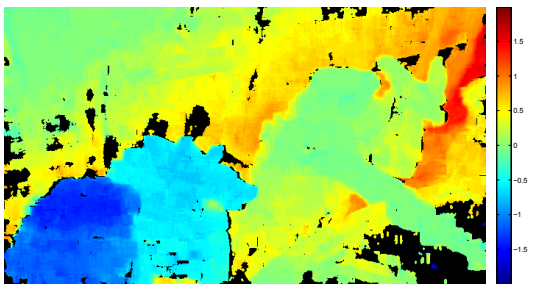
8.4 Image interpolation

The views we can render with our camera have a very low resolution (0.12 megapixels with valid information) and the extraction process we use, either generate inclined images (K-space) or a rectangular image but with half of pixels without any valid information (R-space).

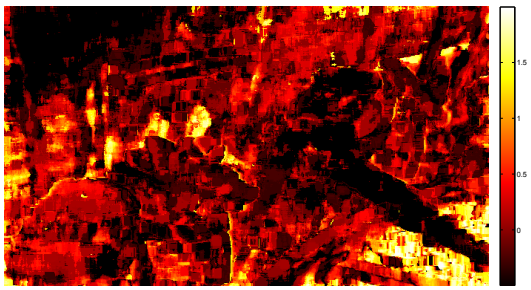
A traditional approach to gain quality on the disparity maps is to execute a subpixel search to estimate the disparity, what implies in interpolating the views before the process.

We propose three interpolation algorithms to reconstruct the missing information of the views (see Figure 8.11) :

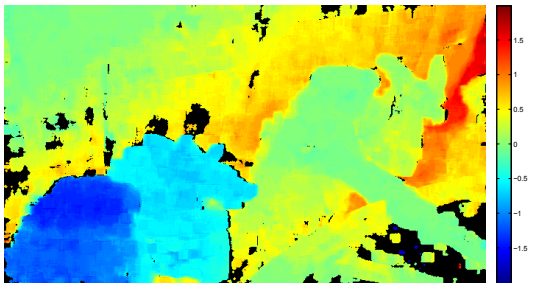
- **Linear:** The simplest and fastest interpolation method, but implies in abrupt jumps on the value of the first derivative of the resulting curve, introducing high frequency terms to the spectrum of the data.
- **Cubic Spline:** A more robust interpolation method [31] that results in continuous first and second derivative, better conserving the spectrum of the data.



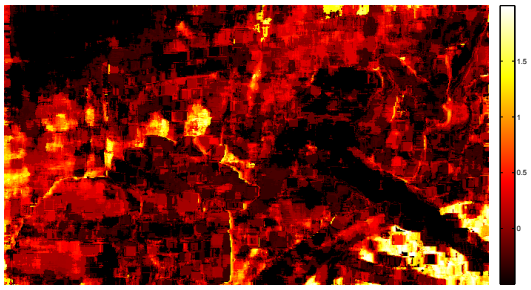
(a) SSD: Estimated disparity



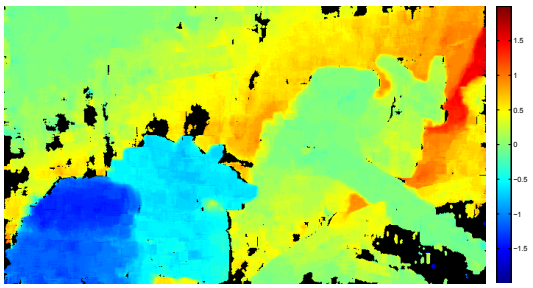
(b) SSD: Error between estimations



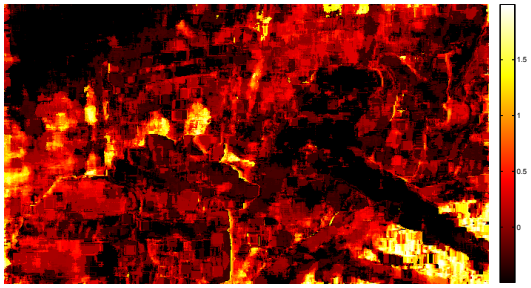
(c) SSD (Vignetting Correction): Estimated disparity



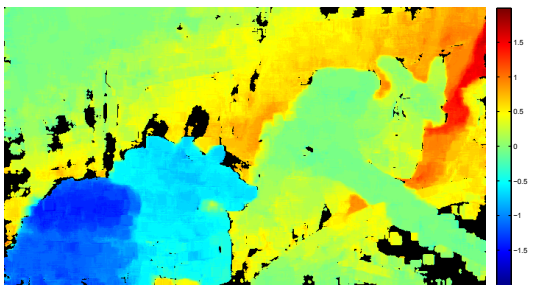
(d) SSD (Vignetting Correction): Error between estimations



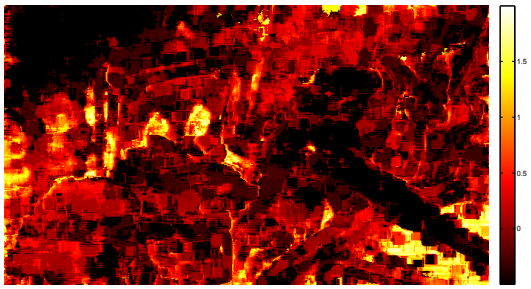
(e) ZSSD: Estimated disparity



(f) ZSSD: Error between estimations



(g) NCC: Estimated disparity



(h) NCC: Error between estimations

Figure 44: Disparity estimation and error with different cost functions for the views in the line $q_y = 3$, (reference view = {3,3}). Black pixels on the disparity maps correspond to regions where the error value is superior to 6 (considered as noisy).

- **PCHIP:** The Piecewise Cubic Hermite Interpolating Polynomial is similar to the cubic spline, but try to preserve the constant characteristic of the data whenever possible, avoiding the oscillating artefacts introduced by the splines when the color of an image changes abruptly from on object to another.

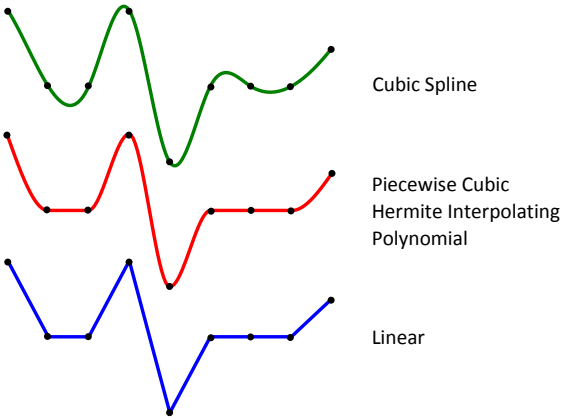


Figure 45: The three interpolation methods we have compared to reconstruct the views represented by their behavior to the given set of values

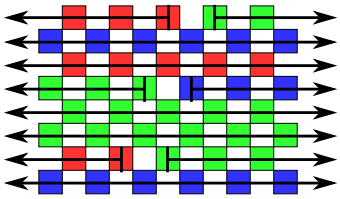


Figure 46: Interpolation lines

The cubic spline and the PCHIP have the same computational cost.

During the interpolation we need to take into account the color pattern of the views to avoid mixing information coming from different color channels. But the fact that the pattern privileges long horizontal sequences of pixels belonging to the same color channel becomes now an advantage.

Hence the interpolation is done only on the horizontal direction and on the regions where we have enough available data to execute the interpolation, as schematised in Figure 8.12, which represents the resulting color pattern on the R-space.

As the images on the R-space have already half of their pixels without information, the interpolation is used as well to reconstruct their values. This implies that just interpolating the missing pixel on the views at the R-space will result in an effective gain on the horizontal resampling by a factor of two, even though that the image still have the same size. We denote by effective resampling the gain we obtain on the number of samples containing valid information and by practical resampling the gain on the total number of samples. Therefore, for images on the K-space the effective and pratical resampling will be equal, but on the R-space the effective resampling will be twice as big as the practical resampling.

We observed that the performance of the three types of interpolation is similar and it is hard to decide which one performs better for our images. A better discrimination can be done by comparing the estimated disparity maps with the ground truth disparity map, but we do not have such information available. To overcome this problem let us use a simpler scene (Figure 8.13) composed by a single textured object, with all points placed at a constant distance of the camera

Figure 47: Scene

Interpolation method	RMSE value	
	mean	weighted mean
Linear	0.0779	0.0843
PCHIP	0.0737	0.0858
Cubic Spline	0.0717	0.0833

Table 5: Estimation RMSE value

that will result in a constant disparity map.

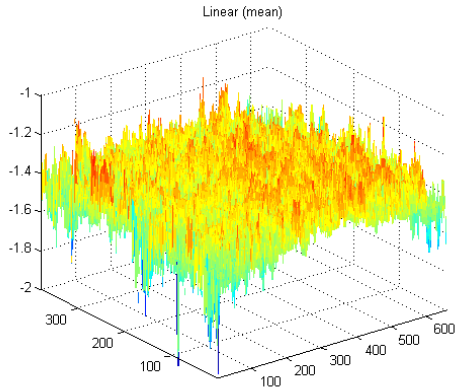
The estimated disparity for this scene is represented on Figure 8.14 for the three types of interpolation. The final disparity is calculated by the mean value of the six estimation and by the weighted mean value based on the dif measure of each cost function.

The mean value of each disparity map can be supposed to be the ground truth. Therefore, the disparity map subtracted by its mean value gives the estimation error. Table 8.1 resumes the Root Mean Square Error (RMSE) value of the estimation error for each interpolation method.

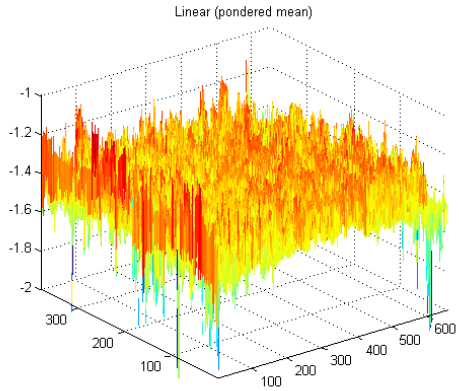
We can observe that the cubic spline is the interpolation with the best performance in terms of the RMSE estimation error, followed by the PCHIP. This result was expected, since the cubic spline is the interpolation, between the three employed, that better preserve the spectrum of the input image and, therefore, better interpolate the texture of the views. Even though, the performance of the PCHIP is not far away from the cubic spline and can better interpolate the transition from one object to another on the views, avoiding oscillations. The performance of the interpolation method is dependent of the input images, but in general the cubic spline performs better than the PCHIP, that performs better than the linear interpolation.

The mean value of the disparity offers a better performance than the weighted mean disparity. This is due to the fact that the quality measure (dif) is related to the texture of the image. Regions well textured will imply in high value of dif for all sets. On the other hand, views distant by 4 usually present higher values of dif than views distant by 2. Therefore the weighted mean privileges the estimation made with views distant by 4 over the estimation of the views distant by 2 even if the second one offers more reliable values than the first one.

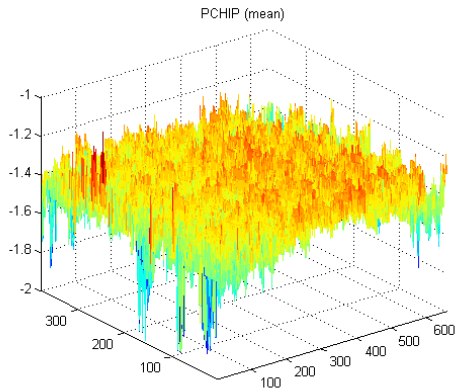
The disparity estimation for the scene on Figure 8.3 is given on Figure 8.15.



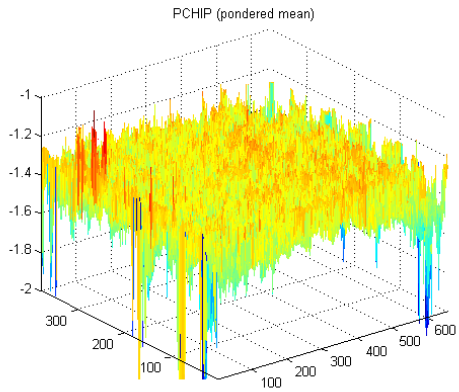
(a) Linear (mean)



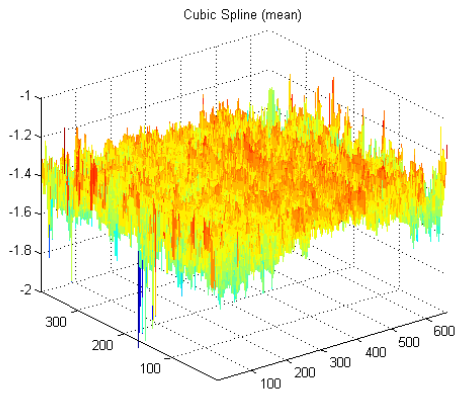
(b) Linear (weighted mean)



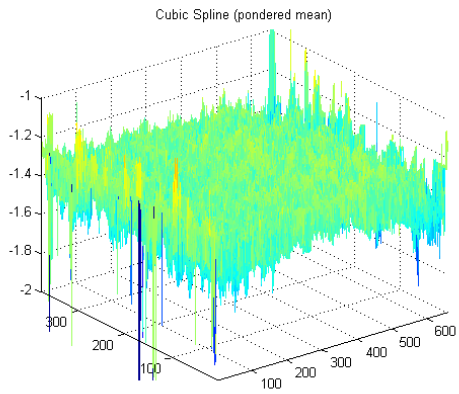
(c) PCHIP (mean)



(d) PCHIP (weighted mean)

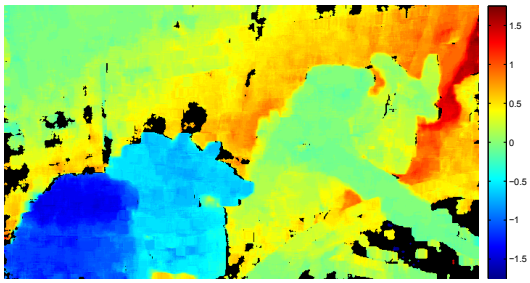


(e) Cubic Spline (mean)

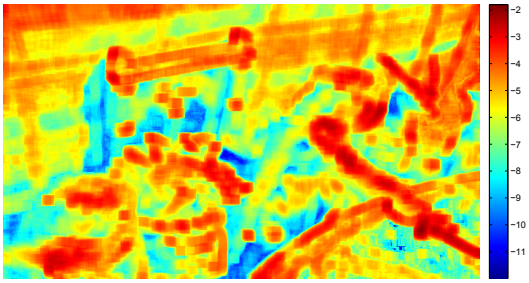


(f) Cubic Spline (weighted mean)

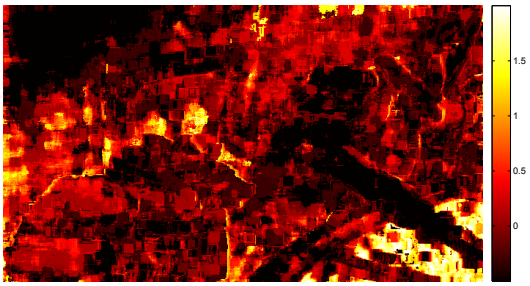
Figure 48: Disparity estimate from photo take at the scene represented in Figure 8.13. The views were interpolated to obtain images with eight times more pixels on the horizontal than the original image. The estimations is performed with a rectangular windows of size 13x51



(a) Disparity estimation



(b) Sum of squared dif



(c) Error

Figure 49: Disparity estimation for the scene on Figure 8.3 using PCHIP interpolation (x8): (a) Disparity estimation map; (b) Sum of squared values of dif for each set of views (log 10 scale). High values correspond to textured regions where we can easily perform the estimation; (c) The error measure, calculated by Equation (8.5). The regions where the error is too high (above 6) were eliminated from the disparity map because they correspond to noise estimation (black regions on the disparity cart)

8.5 Cost function interpolation

Another way to gain precision on the disparity estimation is to interpolate the cost function itself, eliminating the problem introduced when two minimum values of the cost function have the same value (Figure 8.8).

The interpolation of the cost function is usually performed at the same time with a view resampling (by interpolation on the horizontal). The shape of the cost function is represented on Figure 8.16 for different values of resampling.

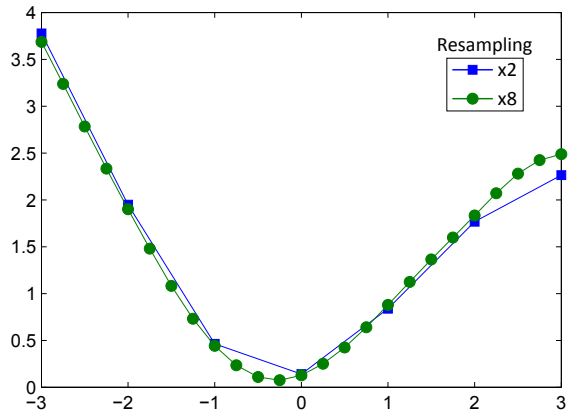


Figure 50: Shape of the cost function for different values of resampling performed by cubic spline interpolation on the horizontal

Based on the shape of the cost function we propose to interpolate using quadratic and cubic polynomials on the neighborhood of the minimum value, as well by using a cubic spline over all points computed for the cost function.

Our interest is to determine the minimum value position of the cost function. Therefore it is not necessary to find a function that cross all the interpolating points of the curve, but one that follow its curvature (smooth interpolation). Thus, for the polynomial interpolations, we employ more than the necessary number of points in a attempt of reducing the noise effect, and the best polynomial p(x), interpolating the points (x_i, y_i) is found as:

$$p(x) = \arg\min_{p(x)} \sum_i [p(x_i) - y_i]^2 w_i^2 \tag{8.9}$$

where w_i is the weight we attribute to each interpolating point (x_i, y_i) that has either a rectangular distribution, equally evaluating the points, or a triangular distribution, making more relevant the points near to the minimum value position obtained by an initial inspection of the discrete cost function values.

We use eight methods to interpolate the Cost Function (CF), summarized in Table 8.2, that differs from function used for interpolation, number of interpolating points and the weight attributed to the interpolating points. For sake of simplification we denote $R_N = \{w_0, w_1, ..., w_{N-1}\}$ the rectangular weight distribution composed by N points with $w_i = 1\ \forall i$ and $T_N = \{w_0, w_1, ..., w_{N-1}\}$ the triangular weight distribution with:

$$w_i = \frac{N+1}{2} - \left| i - \frac{N-1}{2} \right|$$

where $w_{(N-1)/2}$ is the weight attributed to the point of the sampled cost function that is minimal, that is, the minimal value position of the cost function found by a first inspection without executing any cost function interpolation.

Interpolation function	**Number of interpolating points**	**Weight distribution**
Quadratic	3	R_3
	5	R_5
	5	T_5
Cubic	4	R_4
	5	R_5
	7	R_7
	7	T_7
Cubic spline	all	–

Table 6: Cost function interpolation methods

The cost function interpolation increases the resolution of the estimated disparity. In order to perceive the gain of interpolating the cost function we need to use a scene that would result in a high dynamic on the disparity. As before, we don't have the ground truth disparity for ours scenes, thus we use a simple scene that results in a planar disparity (changes linearly with respect to the x and y axis) and we suppose the fit of the disparity by a plan is the ground truth.

Take for example the scene on Figure 8.17. We performed the disparity estimation using the eight methods of the cost function interpolation. By fitting the data with a planar function, which we suppose as the ground truth, we can compute the Root Mean Squared Error (RMSE) value of the estimation. Another standard measure of the quality of the estimated disparity maps can be evaluated by the percentage of Bad Matched Pixels (BMP) [40], where we count the number of pixels that present an estimation error higher than a given threshold. The values we obtain are represented on Table 8.3.

Figure 51: Scene used to test the Cost Function interpolation

From the table we can observe that in terms of RMSE and BMP the quadratic polynomial with a R_3 weight distribution and the cubic spline have the best performance among the cost function interpolation evaluated. On the other hand, the Cubic R_4 interpolation do not reduce significantly the noise compared

	Mean		Weighted mean	
CF interpolation	**RMSE**	**BMP**	**RMSE**	**BMP**
No interpolation	0.3325	38.89%	0.2216	35.64%
Quadratic R_3	0.2374	6.74%	0.3099	29.41%
Quadratic R_5	0.2519	16.49%	0.7137	73.05%
Quadratic T_5	0.2460	12.81%	0.5807	68.15%
Cubic R_4	0.2662	21.25%	0.3336	16.22%
Cubic R_5	0.4537	67.79%	0.5995	73.00%
Cubic R_7	0.5585	73.59%	1.3249	83.40%
Cubic T_7	0.5173	71.50%	0.9564	79.71%
Cubic Spline	0.1860	4.45%	0.2756	22.88%

Table 7: Error of the estimated disparity for different cost function interpolation. The disparity is estimated on line $q_y = 3$ with the view {3,3} as reference and the estimation for each pair is regrouped either by their mean value and by the weighted mean value. The view images employed do not have spacial interpolation. The disparity estimation were computed with the region-based SSD cost function performed on a rectangular windows of size 13x13. To compute the BMP a threshold of 0.2 pixel is employed

to the case of no interpolation and the Cubic R_5, R_7 and T_7 provide disparity maps noisier than the case of no interpolation showing that the cubic polynomial do not adapt well to the cost function. For both quadratic and cubic polynomial interpolation, adding interpolating points reduce the quality of the resulting disparity maps.

We observe as well that, in general, computing the disparity by the weighted mean of the estimated disparities for the six view pairs by the dif value offers a performance below the case of computing the disparity by the simple mean value.

The precision gain we obtain with the cost function interpolation can be easily visualized with the Estimated Disparity Profile Image (EDPI) for the cases where the scene provide a planar disparity.

A disparity map can be seen as a collection of points on a 3D space of the form $(x, y, d(x,y))$, where x, y are the spatial dimension and $d(x,y)$ the disparity value. We will assume the result of Theorem 1 without proof. If the disparity ground truth can be represented as a function of a single variable $g(u)$, $u = \nabla d \cdot (x,y) / \|\nabla d\|$, than the projection of the points $(x, y, d(x,y))$ on the plane containing the gradient of the disparity will result in a line of the form $(u, g(u))$. The EDPI is a map of the density of the projection of these points as schematised on Figure 8.18.

Theorem 1. *Let $f(x,y)$ be a function of $\mathbb{R}^2 \to \mathbb{R}$. Exists $\theta \in \mathbb{R}$, a constant, $-\pi < \theta \leq \pi$, that satisfy:*

$$\|\nabla f\|.(cos(\theta), sin(\theta)) = a.\nabla f; \quad a = \begin{cases} 1 & \text{if } \partial f/\partial x > 0 \\ 1 & \text{if } \partial f/\partial x = 0 \text{ and } \partial f/\partial y \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

If, and only if, $\exists\, g(x)$, a function of $\mathbb{R} \to \mathbb{R}$, that for $u, v \in \mathbb{R}$:

$$f(u.cos(\theta) + v.sin(\theta), u.sin(\theta) - v.cos(\theta)) = g(u)$$

To compute the EDPI we shall need the following result:

Theorem 2. *Let $(u_k, v_k) \in \mathbb{R}^2$ be a sequence of points on the 2D space and $\mu(x,y)$ be the density of points (u_k, v_k), then:*

$$\mu(x,y) = \sum_k \delta(x-u_k)\delta(y-v_k)$$

where $\delta(x)$ is the distribution delta of Dirac

Proof. Let S be a region on the 2D space. $\forall S$:

$$\iint_S \mu(x,y)dxdy = \sum_k \begin{cases} 1 & \text{if } (u_k, v_k) \in S \\ 0 & \text{otherwise} \end{cases}$$

but we know that [30]:

$$\iint_S \delta(x-u_k)\delta(y-v_k)dxdy = \begin{cases} 1 & \text{if } (u_k, v_k) \in S \\ 0 & \text{otherwise} \end{cases}$$

Therefore:

$$\iint_S \mu(x,y)dxdy = \sum_k \iint_S \delta(x-u_k)\delta(y-v_k)dxdy$$

$$\iint_S \mu(x,y)dxdy = \iint_S \sum_k \delta(x-u_k)\delta(y-v_k)dxdy$$

$$\therefore \mu(x,y) = \sum_k \delta(x-u_k)\delta(y-v_k)$$

□

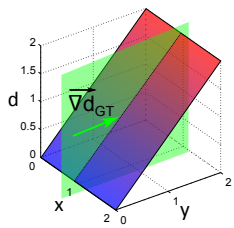


Figure 52: EDPI

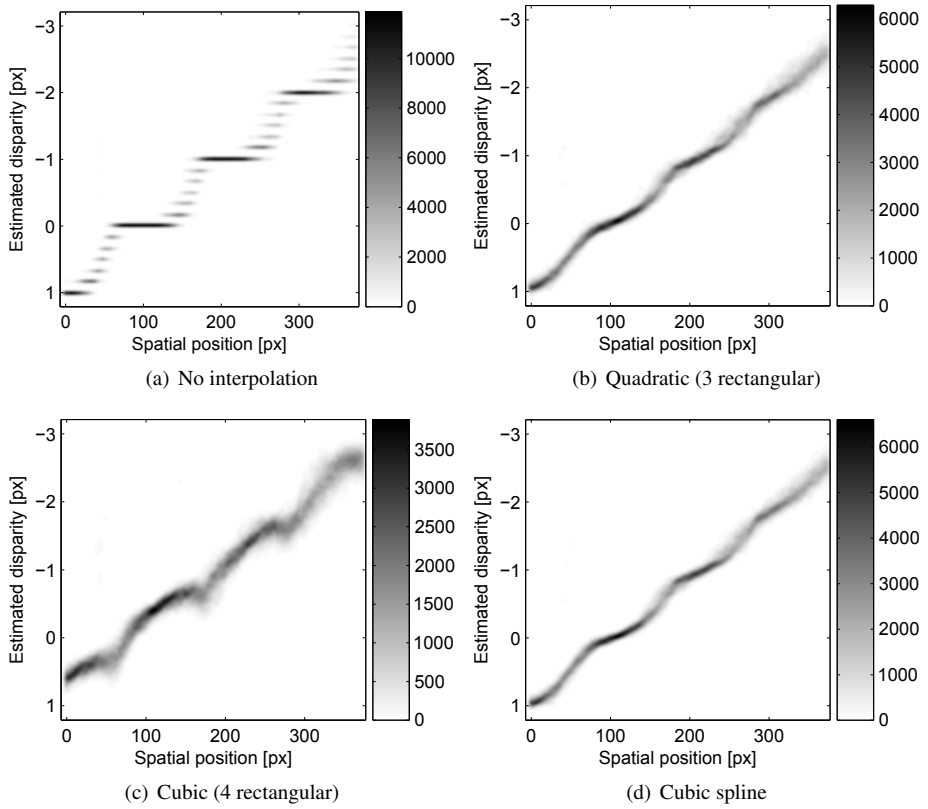
The density of points on the EDPI from the discrete points available are calculate using Theorem 2, but the representation of the density by a sum of Dirac distribution is rather theoretical. Indeed one can not visualize this function because it contains an infinity spectrum. Therefore we apply a low pass filter with a Gaussian shape G_σ to the disparity for sake of visualization and we define the EDPI as below:

$$EDPI(i,j) = G_\sigma(i,j) * \sum_k \delta\left(i - \frac{\overrightarrow{\nabla d_{GT}}}{\left\|\overrightarrow{\nabla d_{GT}}\right\|} \cdot (x_k, y_k)\right) \delta\left(j - d(x_k, y_k)\right)$$

where d_{GT} is the disparity of the ground truth and $\overrightarrow{\nabla d_{GT}}$ its gradient, that is constant for the cases of planar disparity.

The EDPIs for the disparity estimated for different cost function interpolation are shown on Figure 8.19. The gradient vector used to determine the projection plan is calculated from the plan fitted to the data that is supposed to be the ground truth.

As we can see, the case of no interpolation results in a stair effect on the estimated disparity because only discrete values of the disparity can be estimated. Even though we observe a smooth transition from



(a) No interpolation (b) Quadratic (3 rectangular)

(c) Cubic (4 rectangular) (d) Cubic spline

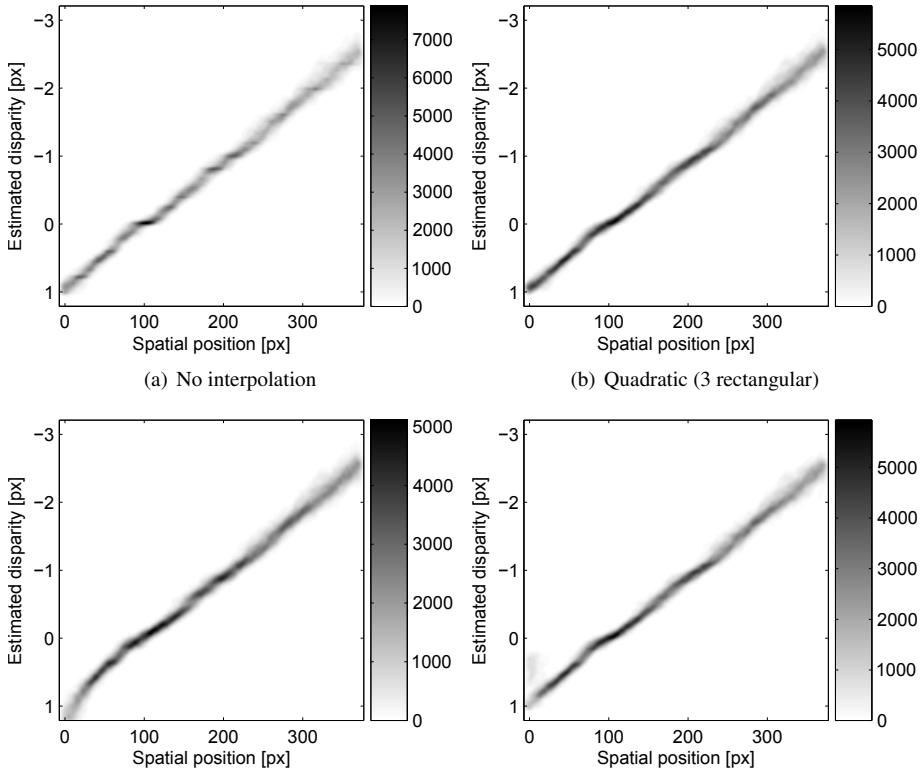
Figure 53: EDPI using different Cost Function interpolation: (a) No cost function interpolation; (b) CF interpolated by quadratic polynomial function using three interpolating points equally weighted; (c) CF interpolated by cubic polynomial function using four interpolating points equally weighted; (d) CF interpolated by cubic spline

one level to another because six values of disparity, estimated from different view pairs, are averaged to obtain the final disparity. On Figure 8.19-(b) and (d) the Cost Function interpolation was able to reduce the stair effect and result in a more precise estimation. Figure 8.19-(c) represents the case of cubic interpolation, that results in disparity maps offering a poor estimation, but still better than the case of no interpolation.

From Table 8.3 we can observe that the cubic interpolations with more than four interpolating points offer results with an estimation error higher that the case of no interpolation, showing that the cubic polynomials are not adapted for CF interpolation. The Quadratic polynomials offer good performance in terms of RMSE and BMP, but their error measures grow as more interpolating points are added, showing that the cost function can be just locally approximated by a quadratic function. The cubic spline presented the best performance in this case, offering the lowest error measures.

The cost function interpolation is intended to be used at the same time as the spatial interpolation, enabling us to obtain high precision disparity maps. These values are used latter for reconstructing the views with a higher resolution, gathering information from neighboring views and reducing the aliasing artifacts, justifying therefore the need of precision on the estimated values.

Figure 8.20 and Table 8.4 summarize the performance of the Cost Function interpolation in cooperation with the spatial interpolation.



(a) No interpolation

(b) Quadratic (3 rectangular)

(c) Cubic (4 rectangular)

(d) Cubic spline

Figure 54: EDPI using spacial interpolation by PCHIP (x8) and different Cost Function interpolation: (a) No cost function interpolation; (b) CF interpolated by quadratic polynomial function using three interpolating points equally weighted; (c) CF interpolated by cubic polynomial function using five interpolating points equally weighted; (d) CF interpolated by cubic spline

The computation of the resulting disparity value from the six estimation by their mean value almost always present better performance than the case of weighted mean. Therefore the second approach is not used for the subsequent estimations.

From the Table 8.4 we can clearly see that the spatial interpolation is more effective than the cost function interpolation to improve the quality of the disparity maps. The cubic spline interpolation for the cost function only offers a better performance than the quadratic interpolation for the cases of no spatial interpolation. When both types of interpolation are executed at the same time the cubic spline and the cubic polynomials produce error measures higher than the case of no cost function interpolation. Table 8.5 reinforces this aspect, being the quadratic polynomials the only type of cost function interpolation that always produce results with error measures lower than the case of no interpolation. This is a evidence that near to the minimum of the cost function the curve presents a shape that can be locally approximated by a quadratic curve. As we move away from the minimum this approximation is not valid anymore, therefore,

Interpolation		Mean		Weighted mean	
Spatial	**Cost function**	**RMSE**	**BMP**	**RMSE**	**BMP**
No interpolation	No interpolation	0.3325	38.89%	0.2216	35.64%
	Quadratic R_3	0.2374	6.74%	0.3099	29.41%
	Cubic Spline	0.1860	4.45%	0.2756	22.88%
	Cubic R_4	0.2662	21.25%	0.3336	16.22%
Linear	No interpolation	0.0977	4.30%	0.1131	6.43%
	Quadratic R_3	0.0893	3.31%	0.1643	9.74%
	Cubic Spline	0.1154	6.16%	0.2011	12.66%
	Cubic R_4	0.1184	8.15%	0.1643	16.78%
Cubic Spline	No interpolation	0.0919	3.42%	0.1073	5.31%
	Quadratic R_3	0.0849	2.69%	0.2029	10.25%
	Cubic Spline	0.1076	4.64%	0.1849	12.66%
	Cubic R_4	0.1102	6.43%	0.1640	15.04%
PCHIP	No interpolation	0.0941	3.59%	0.1094	5.15%
	Quadratic R_3	0.0882	3.04%	0.1614	8.99%
	Cubic Spline	0.1089	4.89%	0.1656	10.01%
	Cubic R_4	0.1169	7.39%	0.1525	13.71%

Table 8: Error of the estimated disparity for different cost function interpolation. The disparity is estimated at the line $q_y = 3$ with the view {3,3} as reference and the estimation for each pair is regrouped either by their mean value and by the Weighted mean value. The disparity estimation were computed with the region-based SSD cost function performed on a rectangular windows of size 13x13. To compute the BMP a threshold of 0.2 pixel is employed

		Mean		Weighted mean	
Cost Function	**CF interpolation**	**RMSE**	**BMP**	**RMSE**	**BMP**
SSD (No vignetting correction)	No interpolation	0.0929	3.51%	0.1072	5.44%
	Quadratic R_3	0.0860	2.79%	0.1980	10.61%
	Cubic Spline	0.1128	5.30%	0.1837	12.92%
	Cubic R_4	0.1116	6.67%	0.1647	15.81%
SSD	No interpolation	0.0919	3.42%	0.1073	5.31%
	Quadratic R_3	0.0849	2.69%	0.2029	10.25%
	Cubic Spline	0.1076	4.64%	0.1849	12.66%
	Cubic R_4	0.1102	6.43%	0.1640	15.04%
ZSSD	No interpolation	0.0876	2.59%	–	–
	Quadratic R_3	0.0797	1.86%	0.0917	2.74%
	Cubic Spline	0.1018	3.66%	0.1384	4.66%
	Cubic R_4	0.1061	5.59%	0.1339	8.93%
NCC	No interpolation	0.0878	2.56%	–	–
	Quadratic R_3	0.0797	1.90%	0.0884	2.32%
	Cubic Spline	0.0838	1.99%	0.0988	2.89%
	Cubic R_4	0.1065	5.65%	0.1333	8.36%

Table 9: Error of the estimated disparity for different cost function algorithms. The disparity is estimated at the line $q_y = 3$ with the view {3,3} as reference and the estimation for each pair is regrouped either by their mean value and by the Weighted mean value. The view's image used for estimation were spatially interpolated by a cubic spline with a zoom factor of 8. The disparity estimation were computed with a rectangular windows of size 13x13. To compute the BMP a threshold of 0.2 pixel is employed

adding more points for the interpolation actually reduce its capacity to precisely determine the minimum position.

Table 8.5 summarizes the performance of the cost function interpolation and the algorithm employed to compute the cost function. The SSD and the SSD with vignetting correction present very similar performance, the second showing better results. This show that the vignetting correction we apply is not able to suppress all the artifacts produced by vignetting, specially because the lens aberrations are ignored and the vignetting approach only treats the effect on the subimages. With better calibration RAW data this characteristics could be estimated more precisely and further improvement is possible.

The ZSSD and the NCC are the cost function algorithms with the best performance, producing very similar error measures. As the ZSSD is less complex in terms of calculus necessary to compute the cost function than the NCC it became the standard algorithm employed for disparity estimation. The cost function interpolation by a quadratic polynomial with three interpolating points and the spatial interpolation with the cubic spline also became an standard approach in view of the quality gain we can obtain on the final result and their performance compared to the other approaches we tested.

8.6 Disparity transposition

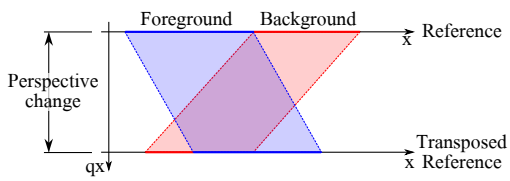


Figure 55: Perspective transposition as seen by the EPI. The blue lines correspond to regions where the disparity value indicate that the object belong to the foreground and the red ones to the background. The transposition process take into account the occlusion.

On Equation 8.2, 8.6, 8.7 and 8.8 we observe that the cost function is dependent of the view chosen as reference. The main effect of the reference view is that its perspective of the scene will reflect on the perspective of the disparity map. In other words, the disparity map will see the scene from the point of view of the reference view. Therefore the disparity map presents parallax and its parallax is the same as that seen on the views.

It comes out that the information available on the disparity can be used to determine how it would be perceived if seen by another perspective without the need of re-executing the estimation algorithm. On this work we denote “disparity transposition” and “view transposition” the change of perspective on the disparity maps and views, respectively, and “perspective transposition” for the general case, since both the disparity and view transposition are similar.

Let $d_R(x,y)$ and $I_R(x,y)$ be the reference disparity and reference view, respectively, $d(x,y)$ and $I(x,y)$ their perspective transposed counterpart and “a” the distance between the perspective to be transposed and the reference perspective, then:

$$Transposition \begin{cases} Disparity & d(x,y) = d_R(x + ad_R(x,y), y) \\ View & I(x,y) = I_R(x + ad_R(x,y), y) \end{cases} \tag{8.10}$$

Figure 8.21 schematizes the perspective transposition. The method has some limitations since regions

on the transposed perspective that are occluded can not be predicted, leaving some holes without information. To overcome this problem one can use two disparities as reference. If one of the reference disparities has the perspective of a view before the perspective of the transposed disparity and the other the perspective of a view after, all the occlusion can be managed and a dense representation can be obtained for the transposed disparity because the information that is occluded in one of the perspectives will not be occluded on the other one. The transposition with two references is schematized on Figure 8.22.

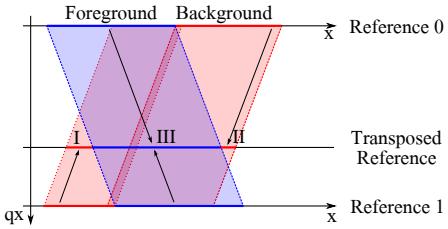


Figure 56: Perspective transposition as seen by the EPI using two disparities to manage the occlusion. I: The information to be transposed was occluded on Reference 0 but available on Reference 1, II: Occluded on Reference 1 but available on Reference 0, III: Not occluded on both references.

Changing the reference view modifies the cost function. Therefore, a region on one disparity that resulted in noise values for the cost function, either by occlusion phenomenon, image noise or aliasing, could result in more reliable values for another disparity differing in perspective since their cost function is not the same. The idea is to use this rich information to try to reduce the noise of a disparity by searching its information in another disparity map.

Due to the color pattern, for instance we are able to perform the disparity estimation only on the horizontal. The approach we propose is, in a given line of views, compute the disparity using the method previous mentioned given by Equations (8.6), (8.7) and (8.8) employing different views as reference. The information of one disparity can be merged with other maps by transposing their disparities to the perspective of interest and comparing their values.

The confidence of the estimated values is determine by thresholding the error map (see Equation (8.5)). When the disparity value of the transposed disparity is reliable but the current value is not, this one is replace by the transposed disparity value. On the other hand, if both values are not reliable we average their values in an attempt to reduce artefacts of the estimation. When both values are reliable we verify if there is any occlusion. Negative values of disparity are related to objects falling between the focal plane of the camera and the main lens and positive values to those falling between the focal plane and the infinity. Therefore if the transposed value is smaller than the current value and is reliable it imply that the disparity value of the transposed disparity is occluding the current value. If the transposed disparity value is bigger, then the current value is occluding the transposed value.

We assume that even though the measures are reliable they have some additive noise. Therefore we introduce a threshold to compare the disparity values and determine if they have the same information. On the cases where the transposed value falls inside the region determined by the disparity threshold the values are supposed to carry the same information and are averaged in an attempt to reduce the estimation noise.

The four possible cases are summarized on Table 8.6. Let e and d be the error and disparity value

of the disparity map where the information will be merged, e_T and d_T the error and disparity value for another disparity on the same line of views but with a different perspective after disparity transposition to the perspective of the disparity of interest, Th_e the threshold we apply to the error value to determine the reliability of an estimation and Th_d the threshold we apply to the disparity values to determine with they have equivalent information:

	$e < Th_e$	$e \geq Th_e$
$e_T < Th_e$	if $d_T < d - Th_d$ Transfer d_T to the cart if $d_T \geq d_T + Th_d$ Do not transfer information otherwise Average values	Transfer d_T to the cart
$e_T \geq Th_e$	Do not transfer information	Average values

Table 10: Rules for disparity merging

Let us take for example the habitual scene. The result of merging the information from disparities computed using as reference the views {3,0}, {3,1}, {3,2}, {3,3}, {3,4}, {3,5}, {3,6} and {3,7} to reduce noise on the disparities with reference view {3,0} and {3,7} is shown on Figure 8.23.

The disparity maps having a reference view near to the edges of the main lens usually are noisier that the ones computed with a reference near to the center. We observe that using this approach enable us to eliminate most of the artefacts, even for the cases of disparity highly corrupted by noise. Even though the disparity on the edges of the main lens are noisier, they can be used to help reduce the artefacts on the disparity near the center because the approach take into account the confidence of the estimation, as we can observe on the disparity with the reference {3,3}.

On the example shown, the region of the disparity on the table is particularly less noisy on the disparity with the reference {0,0} than the ones with the reference {3,3} and {3,0} and could be used to improve even more the quality of the other disparities if we have available the vertical disparity.

We use this approach to merge the information of a sequence of disparity maps and obtain two views on the extreme of a line of views with higher quality, since we observed that only two disparity charts, with the perspective of the views on the extremes of a line of view store enough information to determine the disparity on any other perspective inside this line o views.

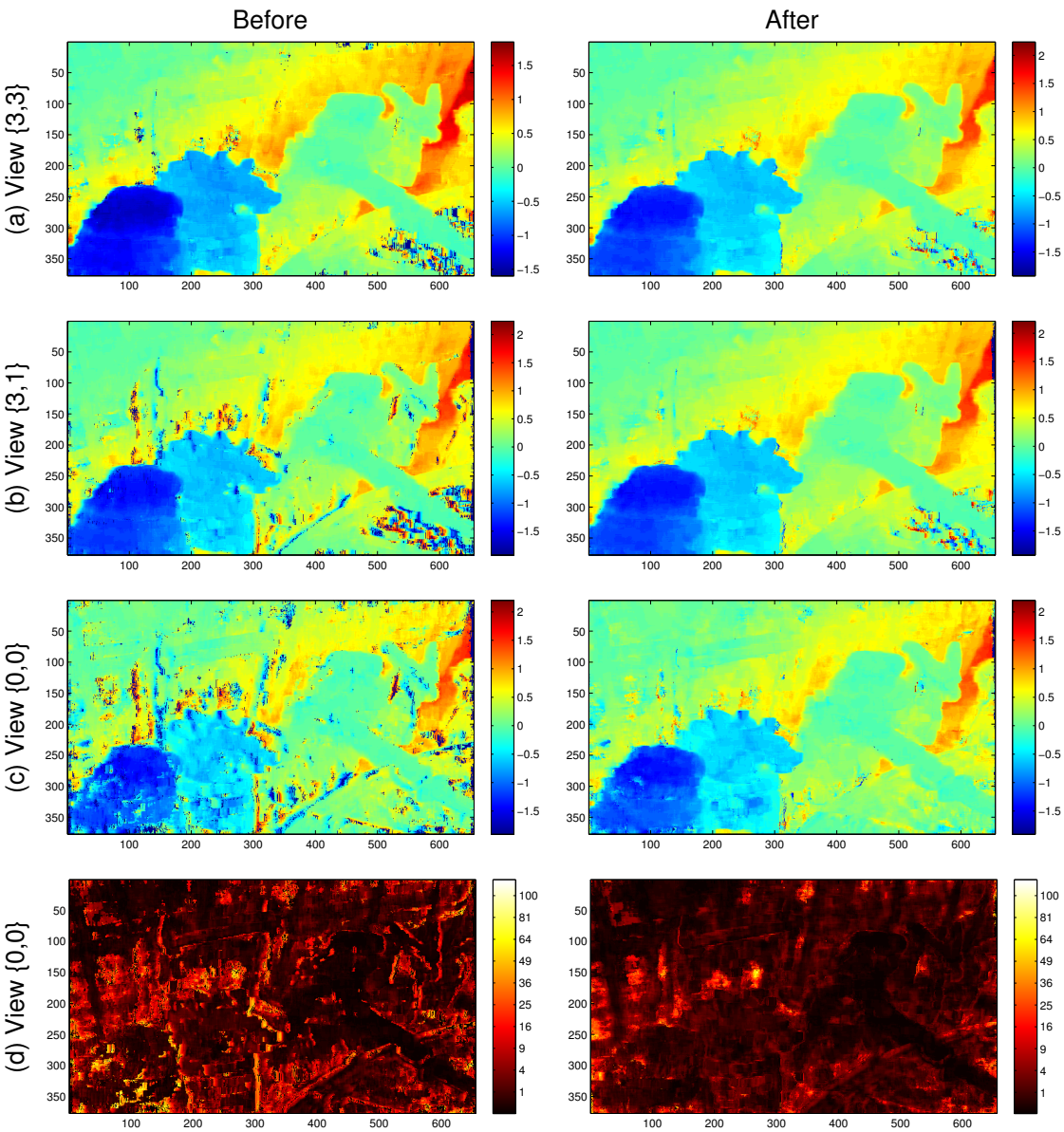


Figure 57: On the first column, the result of estimating the disparity using the ZSSD cost function with a rectangular window of size 13x51 on views with spline spatial interpolation and Quadratic R_3 cost function interpolation. On the second column the resulting of merging the disparity maps of eight different perspective belonging to the same line of views. The thresholds used are $Th_e = 6$ and $Th_d = 1$. In (a), (b) and (c) the disparity maps for three different perspectives, on (d) the error map with the perspective of the view {0,0}.

9 View reconstruction

9.1 Reconstructing the view's image

With the two disparity maps computed as described on Section 8.6 we are now able, for a given view, to merge its pixels with the pixels of its neighboring views and obtain a richer image.

To manage the occlusion, from the two reference disparity we compute the disparity at the perspective of the view of interest (the view to be reconstructed) and the disparity at the perspective of the view we use as base from transposing the pixels (the view from where we will gather information to reconstruct the interest view).

The pixels of the base view are transferred in a four step algorithm, as represented on Figure 9.1:

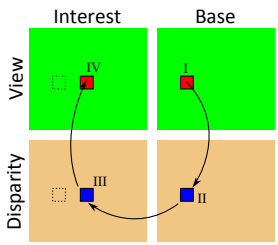


Figure 58: Pixel transfer

- **I:** Swap the base view image to find pixels with valid information
- **II:** Once a valid pixel is found we read on its disparity cart the swap to apply to the pixel position to obtain its equivalent position on the interest view image
- **III:** On the equivalent position of the pixel, we compare the value of the disparity map at the perspective of the interest view to determine if the pixel we are about to transfer is not occluded (an occlusion imply that the disparity on the perspective of the base view is higher than its equivalent value on the perspective of the interest view)
- **IV:** If the pixel is not occluded it is transferred to the interest view image. If the image already has a valid pixel information for the color channel being transferred on this position, their values are averaged.

Figure 9.2 show the resulting color pattern at the reconstructed view {3,3}, using the pixels information from the view {3,3} and {3,4} for the three color channels. Pixels in green are those who already was present on the view image (pixel information coming from view {3,3}) and pixel in red are those merged to the image due to the transposition of the information from view {3,4} to view {3,3}.

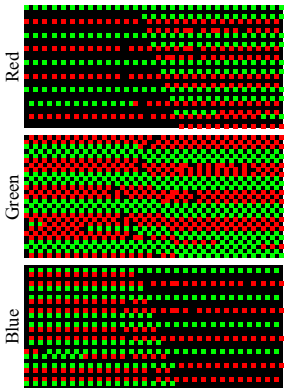


Figure 59: Color Pattern

The green channel result in a very rich information well distributed over the entire image but for the red and blue channels we observe a transition from regions with rich information to poor information. This is due to the Bayer pattern on the subimages. If we take a line (or a column) of pixels on the Bayer pattern we observe that it is composed uniquely by two color (blue-green or green-red). Therefore, views displaced on the horizontal will have poor information of red and/or blue on the same regions and the reconstruction of the views will not be capable of overcoming this effect unless we have available the information about the vertical disparity.

As the lines on the Bayer pattern change from blue-green to green-red and vice-verse from one line to another, reconstructed views distant by an even number on the vertical will have poor/rich information from the red/blue channel on the same regions. Therefore, the same way as we did for the horizontal disparity estimation before demosaicing, we can estimate the disparity on the vertical between views distant by 2.

Thus, we use as cost function for disparity estimation on the vertical the symmetric of the equations used for horizontal disparity:

- **Sum of Squared Differences (SSD):**

$$CF_{x_0,y_0}(d) = \frac{\sum_{i,j\in\Omega} K\left[I_n(x_i,y_j+a_nd)-I_m(x_i,y_j+a_md)\right]^2}{\sum_{i,j\in\Omega} K} \tag{9.1}$$

- **Zero-Mean Sum of Squared Differences (ZSSD):**

$$CF_{x_0,y_0}(d) = \frac{\sum_{i,j\in\Omega} K\left[I_n(x_i,y_j+a_nd)-\tilde{I}_n-I_m(x_i,y_j+a_md)+\tilde{I}_m\right]^2}{\sum_{i,j\in\Omega} K} \tag{9.2}$$

- **Normalized Cross Correlation (NCC):**

$$CF_{x_0,y_0}(d) = \frac{-\sum_{i,j\in\Omega} K\left[I_n(x_i,y_j+a_nd)-\tilde{I}_n\right]\left[I_m(x_i,y_j+a_md)-\tilde{I}_m\right]}{\sqrt{\sum_{i,j\in\Omega} K\left[I_n(x_i,y_j+a_nd)-\tilde{I}_n\right]^2}\sqrt{\sum_{i,j\in\Omega} K\left[I_m(x_i,y_j+a_md)-\tilde{I}_m\right]^2}} \tag{9.3}$$

The view images used for the estimation are reconstructed on each view's line merging the information from all the views having the same q_y value. As the number of invalid pixels is very small on these cases (except for the regions with poor information for the red/blue channel) we decided to apply a demosaicing to obtain more pixels to compare on our region-based research of similarity between the pair of images.

The demosaicing we apply is a bi-linear interpolation that do not suppose any color pattern for the images. The algorithm, for each point with not valid information (unknown intensity value), search the four nearest neighbors of the pixel to be interpolated containing known values of intensity. Whenever not four neighbors are found on the neighborhood of the pixel its value is not interpolated and left unknown (or invalid).

As most classical approaches of demosaicing, we first interpolated the green channel, supposing that it is the most rich of all the other channels. Then, to take advantage of the spectral correlations, we interpolate the red-green and blue-green differences images, called primary difference signals (PDS), rather than directly recovering the missing color samples. The interpolated PDS and green channel are then used to recover the red/blue intensity.

The performance of the vertical disparity estimation computed by this approach is shown in Figure 9.3 compared to the horizontal disparity. We can clearly observe that the vertical disparity is noisier than the horizontal since the view images passed through several processing before entering the disparity estimation algorithm (vignetting correction, view reconstruction based on horizontal disparity and demosaicing), what can introduce some artifacts.

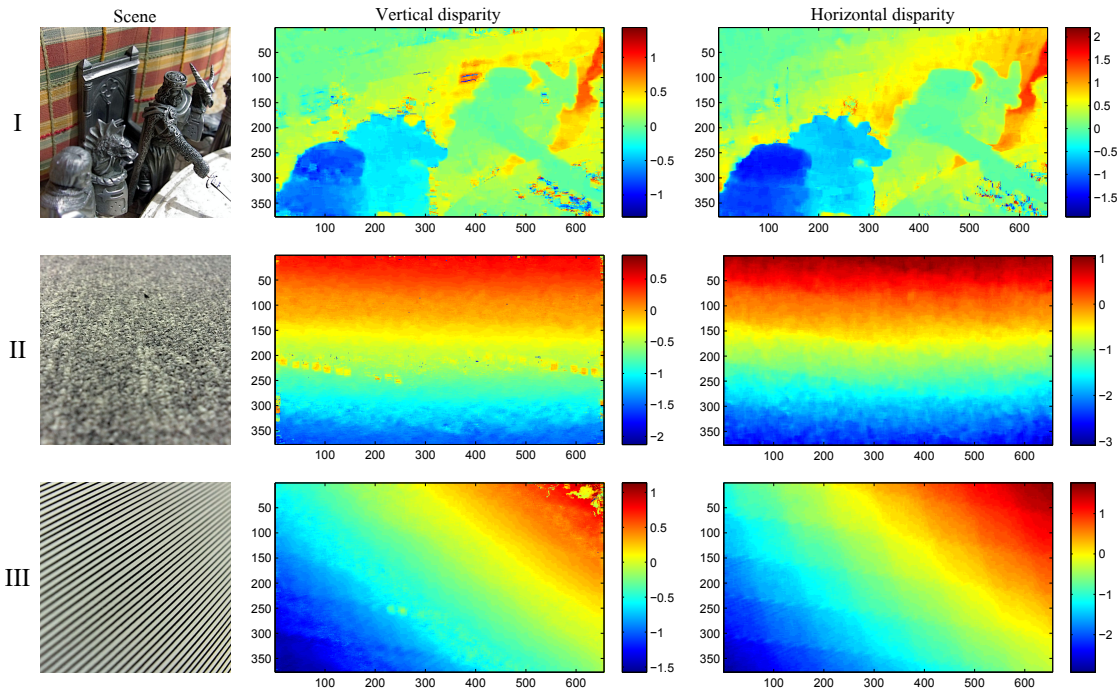
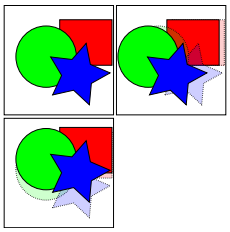


Figure 60: Vertical and horizontal estimated disparity for the scenes on the first column using as reference the view {3,3}. The horizontal disparity was estimated as described on Chapter 8 and the vertical disparity by the method described in this section. Both horizontal and vertical disparity were merged with 8 disparity carts to reduce the noise.



However one can observe the similarity between the horizontal and vertical disparity. With we take for example the schema on the right we could raise the hypotheses that there is a relationship between the parallax on the horizontal and vertical. We observed that objects on the scene falling on the focal plane do not present horizontal parallax because, as its virtual image focalise on the microlens array, all the light rays entering the camera coming from the same position on the object will fall on the same microlens. Therefore we could expect that a disparity equal to zero on the horizontal would imply on a zero disparity on the vertical as well.

To verify this relationship we can analyze the density of point (dv, dh), where dv is the vertical disparity and dh the horizontal disparity, on the plan dv versus dh. The density is compute using Theorem 2 and a Gaussian low pass filter for visualization. The density functions are shown in Figure 9.4.

We can observe that the relationship between the horizontal and vertical disparity is linear and the parameters of the line containing the highest density of points are similar for all scenes. Indeed, by fitting the data with a linear polynomial, we found the following equation, valid for all scenes:

$$dh = \sqrt{3}dv \tag{9.4}$$

This result could be expected since the views created on the R-space that where used for disparity estimation have different sampling on the horizontal and vertical. To try to copy the displacement of the

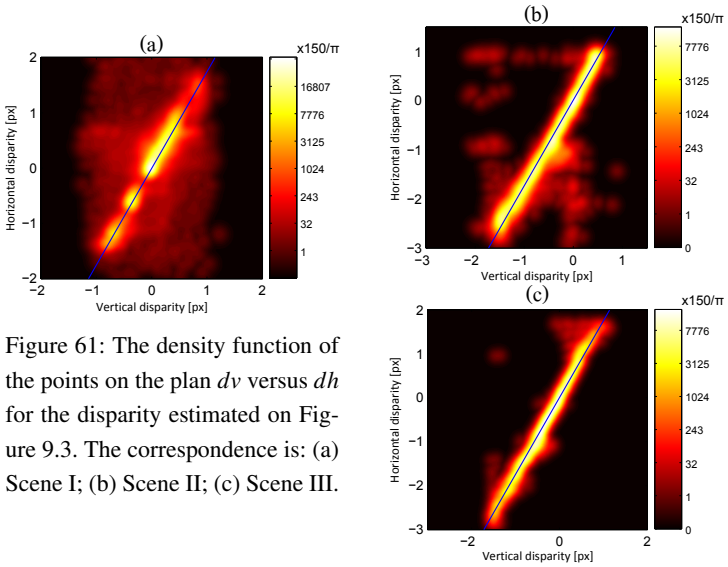


Figure 61: The density function of the points on the plan dv versus dh for the disparity estimated on Figure 9.3. The correspondence is: (a) Scene I; (b) Scene II; (c) Scene III.

subimages on the RAW data we employed an horizontal sampling $\sqrt{3}$ higher than the vertical sampling for rendering the view's image.

Mathematically, this result can be proved by finding the relationship between depth and disparity. On Annex A, supposing that the vertical and horizontal sampling are equal, we found the following equation:

$$d_h = \frac{z_0 f^2}{n\delta(z_0 - f)^2} \frac{z_0 - z}{z} \tag{9.5}$$

$$d_v = \frac{z_0 f^2}{n\delta(z_0 - f)^2} \frac{z_0 - z}{z} \tag{9.6}$$

where z_0 is the depth of the focal plan of the camera, f its focal distance, δ the distance between the microlens array and the camera sensor, n the diameter of the microlens in number of pixels and z the depth of an object on the scene.

This relationship enable us to determine the parallax on the vertical without the need to estimate the vertical disparity. We can use this result to take profit of our richer information to execute the view and disparity merging using the images an maps displaced on the vertical as well. Figure 9.5 compares the disparity as we obtain by minimizing the cost function and the maps reconstructed merging horizontal and vertical distant maps.

The disparity map with reference at {3,3} present less noise than the map with reference at {0,0} and even a disparity merging only with horizontal maps is capable of significantly reducing the artifacts, except for the region on the table that was a noisy region on all disparity maps on the horizontal. For the disparity with reference at {0,0} this region is less noisy, therefore the disparity merging taking into account all disparity maps is capable of transferring this information to produce a better result at the reference {3,3}.

The disparity at {0,0} gains significantly quality with the use of the vertical information but we can

observe that on the region on the table, comparing the disparity merged using only horizontal views, appeared some artifacts. These artifacts are introduced by erroneous estimation of the reliability of an estimation, that is done by thresholding the error map.

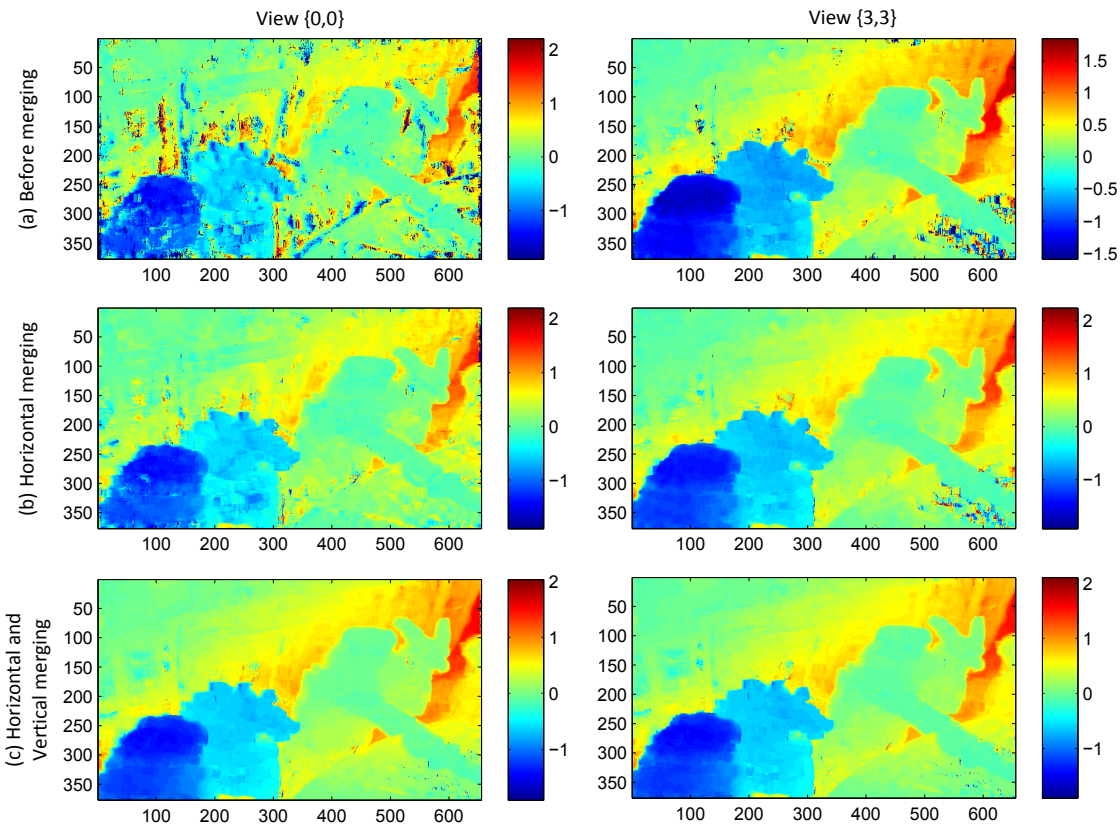


Figure 62: Disparity maps for the reference views computed using the ZSSD algorithm, with a quadratic R_3 cost function interpolation and spline view interpolation by an effective factor of x4 and a rectangular windows of size 13x51: (a) The resulting map, without disparity merging; (b) The reconstructed disparity maps merging the 8 maps present on the same line of views; (c) The reconstructed disparity maps merging the 64 disparity maps using the horizontal disparity to predict vertical parallax.

We can expect that changing the decision threshold would imply in different results. To test the performance of the disparity merging we use a scene that result in a planar disparity map, as before, but containing regions not textured to introduce noise to the estimation (see Figure 9.6).

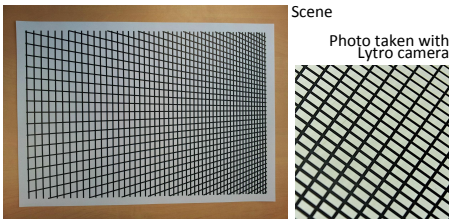


Figure 63: Employed scene

Figure 9.7 show the RMSE and BMP for the disparity maps after horizontal and vertical merging as a function of the error threshold. Using a high value for the threshold can make us consider reliable the noise estimation, reducing the quality of the resulting maps. From the curve we can see that the RMSE and the BMP usually grow when we grow the threshold error and tends to stabilize for high values of threshold.

On the other hand, using a low value for the error threshold can make us consider unreliable the estimation that actually are reliable, and the algorithm will therefore average its values with noise estimations in an attempt to reduce artifacts. This effect is most evident on the BMP curve.

From the curves we conclude that using an error threshold around 1 offers a good compromise between the error of considering unreliable the reliable estimation and considering reliable the noise estimations for this image.

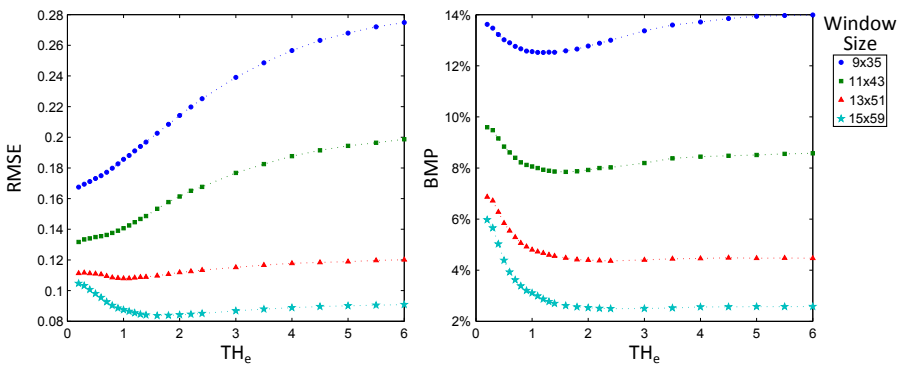


Figure 64: Error measures on the disparity estimation after disparity merging as a function of error threshold adopted. All the disparity estimation were execute with the ZSSD algorithm, quadratic R_3 cost function interpolation, spline spatial view resampling by an effective factor of x8 (x4 in practice) and using rectangular windows of the indicated sizes.

Table 9.1 and 9.2 compare the estimation error before and after the disparity merging for one disparity map with reference near to the board of the main lens and another near to the center. After the merging both disparity maps have an similar estimation error, showing that the view merging equalize the quality of the estimated disparity maps for all positions.

Window size	{0,0}		{3,3}	
	Before	**After**	**Before**	**After**
9x35	0.3940	0.1856	0.2415	0.1860
11x43	0.3588	0.1407	0.1896	0.1414
13x51	0.3270	0.1079	0.1361	0.1094
15x59	0.3101	0.0875	0.1005	0.0886

Table 11: RMSE before and after disparity merging using $Th_e = 1$

Window size	{0,0}		{3,3}	
	Before	**After**	**Before**	**After**
9x35	68.90%	12.55%	21.73%	12.60%
11x43	66.57%	8.06%	14.09%	8.17%
13x51	63.97%	4.80%	7.77%	4.96%
15x59	63.41%	3.11%	3.39%	3.24%

Table 12: BMP before and after disparity merging using $Th_e = 1$

Conclusion

In this work we have presented a method to estimate the 3D depth map of a scene from plenoptic images. Unlike an array of cameras, the light field is captured by a single device, adding complexity for view rendering because of the elevated number of microlenses and their disposition as well for demosaicing the images due to the resulting color pattern on the rendered views. The proposed region-based cost function adapted to views not demosaiced showed that the disparity estimation can be performed resulting in satisfying disparity maps.

The configuration of the camera imply in a relationship between disparity and depth different from those of multi-view stereo, but the inverse proportionality with respect to the depth is found in both approaches. Indeed plenoptic cameras share most of the caracteristics of the classical multi-view stereo approach and we can extrapolate methods used on those cases with little adaptation, such as view and cost function interpolation.

The error map produced showed to be a robust measure to determine the reliability of the estimated disparity by identifying the artifacts generated by image noise, not textured regions and aliasing. This artifacts can be eliminated or attenuated by merging the information computed with different perspectives using the error map to control the information flow from one disparity map to another.

These disparity maps can be used to reconstruct the missing information on the views deriving on a demosacing approach assisted by the disparity measure. This gives perspectives for the future to use super-resolution algorithms to produce views of the scene with higher resolution and use this new images to estimate the disparity with a better precision and eliminating or reducing the problem of aliasing. We observe that this is a chicken-egg problem: to execute the super-resolution algorithm is necessary to know the disparity and to estimate the disparity with a good precision and without aliasing artifacts is necessary to execute a super-resolution algorithm. On this perspective, this work offered an approach to determine an initial estimation of the disparity to be latter employed on an iterative disparity estimation.

Bibliography

[1] Edward H. Adelson and John Y. A. Wang. Single lens stereo with a plenoptic camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):99–106, February 1992.

[2] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 434–441, 1998.

[3] B. E. Bayer. Color imaging array, July 1976.

[4] TE Bishop and P. Favaro. Plenoptic depth estimation from multiple aliased views. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1622–1629. IEEE, 2009.

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

[6] Sylvie Chambon and Alain Crouzil. Evaluation et comparaison de mesures de corrélation robustes aux occultations. Technical Report 2002 - 34 -R, IRIT, 2002.

[7] Douglas Fredwill Winnek Coffey. Apparatus for making a composite stereograph, December 1936.

[8] Antonio Criminisi, Sing Bing Kang, Rahul Swaminathan, Richard Szeliski, and P. An. Extracting layers and analyzing their specular properties using epipolar-plane-image analysis. Technical report, Computer Vision and Image Understanding, 2002.

[9] Yu. A. Dudnikov. Autostereoscopy and integral photography. *OPTICAL TECHNOLOGY*, 37(7):422–426, July 1970.

[10] Olivier Faugeras, Thierry Viéville, Eric Theron, Jean Vuillemin, Bernard Hotz, Zhengyou Zhang, Laurent Moll, Patrice Bertin, Herve Mathieu, Pascal Fua, Gérard Berry, and Catherine Proy. Real-time correlation-based stereo : algorithm, implementations and applications. Rapport de recherche RR-2013, INRIA, 1993.

[11] Pascal Fua. Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities. In *In Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 1292–1298, 1991.

[12] Andrea Fusiello and Vito Roberto. Efficient stereo with multiple windowing. In *In CVPR*, pages 858–863. IEEE Computer Society Press, 1997.

[13] Todor Georgeiv, Ke Colin Zheng, Brian Curless, David Salesin, Shree Nayar, and Chintan Intwala. Spatio-angular resolution tradeoff in integral photography. In *In Eurographics Symposium on Rendering*, pages 263–272, 2006.

[14] Todor Georgiev. An analysis of color demosaicing in plenoptic cameras. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 901–908, Washington, DC, USA, 2012. IEEE Computer Society.

[15] Todor Georgiev, Chintan Intwala, Sevkit Babakan, and Andrew Lumsdaine. Unified frequency domain analysis of lightfield cameras. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 224–237, Berlin, Heidelberg, 2008. Springer-Verlag.

[16] Dan B Goldman and Jiun-Hung Chen. Vignette and exposure calibration and compensation. In *The 10th IEEE International Conference on Computer Vision*, pages 899–906, Oct. 2005.

[17] N. Grammalidis and M.G. Strintzis. Disparity and occlusion estimation in multiocular systems and their coding for the communication of multiview image sequences. *Circuits and Systems for Video Technology, IEEE Transactions on*, 8(3):328–344, 1998.

[18] M. Harris. Focusing on everything. *Spectrum, IEEE*, 49(5):44–50, 2012.

[19] H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[20] Stephen S. Intille and Aaron F. Bobick. Disparity-space images and large occlusion stereo. pages 179–186. Springer-Verlag, 1994.

[21] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 297–306, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[22] Heung-Yeop Jang, Jin-Ho Ahn, Je-Ho Lee, Yong-Moo Kwon, Sangkuk Kim, and Sang-Hui Park. Fast interpolation technique on epipolar plane image using phase correlation. In *Circuits and Systems, 1997. ISCAS '97., Proceedings of 1997 IEEE International Symposium on*, volume 2, pages 1425–1428 vol.2, 1997.

[23] T. Kanade, H. Kano, S. Kimura, A. Yoshida, and K. Oda. Development of a video-rate stereo machine. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 95–100 vol.3, 1995.

[24] Leonardo da Vinci. The notebooks of Leonardo da Vinci, translated by Jean Paul Richter. Seattle, , circa 1500.

[25] Chia K. Liang, Tai H. Lin, Bing Y. Wong, Chi Liu, and Homer H. Chen. Programmable aperture photography: multiplexed light field acquisition. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, New York, NY, USA, 2008. ACM.

[26] G. Lippmann. Épreuves réversibles donnant la sensation du relief. *J. Phys. Theor. Appl.*, 7(1):821–825, 1908.

[27] Andrew Lumsdaine and Todor Georgiev. The focused plenoptic camera. In *In Proc. IEEE ICCP*, pages 1–8, 2009.

[28] Ren Ng. *Digital light field photography*. PhD thesis, Stanford, CA, USA, 2006. AAI3219345.

[29] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. Technical report, April 2005.

[30] Levent Onural. Impulse functions over curves and surfaces and their applications to diffraction. *Journal of Mathematical Analysis and Applications*, 322(1):18 – 27, 2006.

[31] Katsuyuki V. Ooyama. The cubic-spline transform method: Basic definitions and tests in a 1d single domain. *Monthly Weather Review*, 130:2392–2415, October 2002.

[32] V. Papadimitriou and T.J. Dennis. Epipolar line estimation and rectification for stereo image pairs. *Image Processing, IEEE Transactions on*, 5(4):672–676, 1996.

[33] Nirav Patel. Reverse engineering the lytro .lfp file format, 2012.

[34] F. Perez, A. Perez, M. Rodriguez, and E. Magdaleno. Fourier slice super-resolution in plenoptic cameras. In *Computational Photography (ICCP), 2012 IEEE International Conference on*, pages 1–11, 2012.

[35] F. Perez Nava and J.P. Luke. Simultaneous estimation of super-resolved depth and all-in-focus images from a plenoptic camera. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009*, pages 1–4, 2009.

[36] Christien Perwass and Lennart Wietzke. The next generation of photography, 2010.

[37] S. Roy, J. Meunier, and null. Cylindrical rectification to minimize epipolar distortion. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 393–399, 1997.

[38] Alexander A. Sawchuk. Real-time correction of intensity nonlinearities in imaging systems. *IEEE Trans. Comput.*, 26(1):34–39, January 1977.

[39] Ashutosh Saxena, Jamie Schulte, and Andrew Y. Ng. Depth estimation using monocular and stereo cues. In *In IJCAI*, 2007.

[40] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140, 2001.

[41] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, pages 131–140, 2001.

[42] Peter-Pike Sloan, Michael F. Cohen, and Steven J. Gortler. Time critical lumigraph rendering. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, pages 17–ff., New York, NY, USA, 1997. ACM.

[43] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *Image Processing, IEEE Transactions on*, 3(5):625–638, 1994.

[44] Kuk-Jin Yoon and In-So Kweon. Adaptive support-weight approach for correspondence search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):650–656, 2006.

[45] Wonpil Yu. Practical anti-vignetting methods for digital cameras. *Consumer Electronics, IEEE Transactions on*, 50(4):975–983, 2004.

A Deduction of the disparity equation as a function of the depth

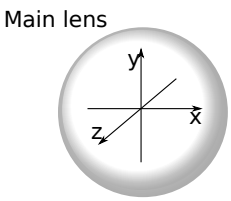


Figure 65: Reference of coordinate system

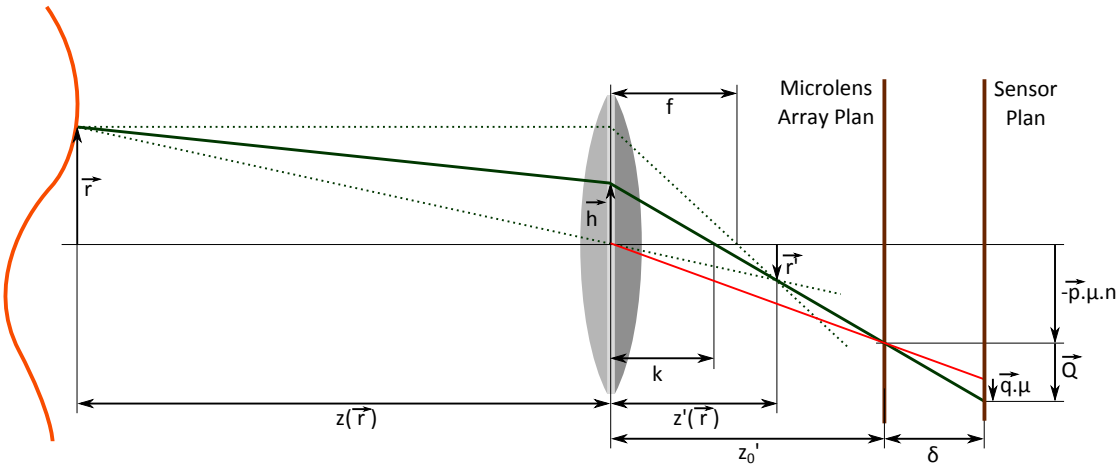
The relationship between the disparity and the depth of an object on the scene is deducted based on the image below. The reference of the coordinate system that we adopt is placed at the camera main lens, as illustrated by Figure A.1. Positive values of z corresponds to points outside of the camera and negative values to points inside.

We suppose that the object captured by the camera is a generic surface at the 3D space, defined by the set of points $(\overrightarrow{r}, z(\overrightarrow{r}))$. We develop our equation to one point of the surface, but the equation is valid for all points.

The Lytro camera focus the object on the scene at the microlens array plan. The focal plan of the camera (plan outside the camera that implies in objects focused at the microlens array plan) is at a distance z_o of the main lens. The distances z_o and z'_o are linked by the thin lens equation:

$$\frac{1}{z_o} + \frac{1}{z'_o} = \frac{1}{f}$$

The size of pixel at the sensor is μ, the diameter of the microlens is $\mu.n$ and the light field captured by the camera is $\mathfrak{R}(\overrightarrow{p}, \overrightarrow{q})$ (the radiance) where $\overrightarrow{p}$ and $\overrightarrow{q}$ are the spatial and angular position of a light ray, respectively, as represented on the figure below. One view of the scene is extracted directly from the radiance function by fixing a value for the angular position, that is: $V_{\overrightarrow{q}}(\overrightarrow{p}) = \mathfrak{R}(\overrightarrow{p}, \overrightarrow{q})$, where $V_{\overrightarrow{q}}$ is the view image. The mathematics representation is deduced from triangle similarity.



A.1 Triangle Similarity

$$\frac{\overrightarrow{h}}{k} = \frac{\overrightarrow{r'}}{z'(\overrightarrow{r})-k} \tag{A.1}$$

$$\frac{\overrightarrow{r'}}{z'(\overrightarrow{r})} = \frac{\overrightarrow{r}}{z(\overrightarrow{r})} \tag{A.2}$$

$$\frac{-\overrightarrow{p}\mu n}{z'_0-k} = \frac{\overrightarrow{h}}{k} \tag{A.3}$$

$$\frac{\overrightarrow{Q}}{\delta} = \frac{-\overrightarrow{p}\mu n}{z'_0-k} \tag{A.4}$$

$$\frac{\overrightarrow{Q}-\overrightarrow{q}\mu}{\delta} = \frac{-\overrightarrow{p}\mu n}{z'_0} \tag{A.5}$$

$$\frac{\overrightarrow{h}}{z'_0} = \frac{\overrightarrow{q}\mu}{\delta} \tag{A.6}$$

$$\frac{1}{z(\overrightarrow{r})} + \frac{1}{z'(\overrightarrow{r})} = \frac{1}{f} \tag{A.7}$$

we define:

$$\overrightarrow{h} = (h_x, h_y); \quad \overrightarrow{r} = (r_x, r_y);$$

$$\overrightarrow{r'} = (r'_x, r'_y); \quad \overrightarrow{p} = (p_x, p_y);$$

$$\overrightarrow{q} = (q_x, q_y) \quad \overrightarrow{Q} = (Q_x, Q_y)$$

. For simplification, we are going to solve the equation only for x, but the result is symmetrical for y.

From Equations (A.2) and (A.7):

$$r'_x = r_x \frac{z(\overrightarrow{r})f}{z(\overrightarrow{r})-f}\frac{1}{z(\overrightarrow{r})}$$

$$r'_x = \frac{r_x f}{z(\overrightarrow{r})-f} \tag{A.8}$$

From Equations (A.8), (A.1) and (A.7):

$$\frac{h_x}{k} = \frac{r_x f}{z(\overrightarrow{r})-f}\frac{1}{\frac{z(\overrightarrow{r})f}{z(\overrightarrow{r})-f}-k}$$

$$\frac{h_x}{k} = \frac{r_x f}{z(\overrightarrow{r})f-k(z(\overrightarrow{r})-f)}$$

$$h_x z(\overrightarrow{r})f - kh_x(z(\overrightarrow{r})-f) = r_x f k$$

$$k(r_x f + h_x z(\overrightarrow{r}) - h_x f) = h_x f z(\overrightarrow{r})$$

$$k = \frac{h_x f z(\overrightarrow{r})}{r_x f + h_x z(\overrightarrow{r}) - h_x f} \tag{A.9}$$

From Equations (A.4) and (A.5):

$$\frac{Q_x}{\delta} = -\frac{p_x\mu n}{z'_0} + \frac{q_x\mu}{\delta} = -\frac{p_x\mu n}{z'_0-k}$$

Adding the results of Equations (A.9) and (A.7):

$$-\frac{p_x\mu n}{z_0 f}(z_0-f) + \frac{q_x\mu}{\delta} = -\frac{p_x\mu n}{\frac{z_0 f}{z_0-f} - \frac{h_x f z(\overrightarrow{r})}{(r_x f + h_x z(\overrightarrow{r}) - h_x f)}}$$

$$-\frac{p_x\mu n}{z_0 f}(z_0-f) + \frac{q_x\mu}{\delta} = \frac{-p_x\mu n(z_0-f)(r_x f + h_x z(\overrightarrow{r}) - h_x f)}{z_0 f^2 r_x + h_x z(\overrightarrow{r}) z_0 f - h_x z_0 f^2 - h_x z(\overrightarrow{r}) z_0 f + h_x z(\overrightarrow{r}) f^2}$$

$$-\frac{p_x\mu n(z_0-f)}{z_0 f} + \frac{q_x\mu}{\delta} = \frac{-p_x\mu n(z_0-f)(r_x f + h_x z - h_x f)}{f^2(r_x z_0 + h_x z(\overrightarrow{r}) - h_x z_0)}$$

$$\frac{q_x}{\delta} = p_x n \left[\frac{z_0 - f}{z_0 f} - \frac{(z_0 - f)(r_x f + h_x z(\overrightarrow{r}) - h_x f}{f^2(r_x z_0 + h_x z - h_x z_0)} \right]$$

$$(\mathrm{I}) = \frac{z_0 - f}{z_0 f} - \frac{(z_0 - f)(r_x f + h_x z(\overrightarrow{r}) - h_x f}{f^2(r_x z_0 + h_x z - h_x z_0)}$$

$$(\mathrm{I}) = \frac{(z_0 - f) f (r_x z_0 + h_x z(\overrightarrow{r}) - h_x z_0) - z_0 (z_0 - f)(r_x f + h_x z(\overrightarrow{r}) - h_x f)}{z_0 f^2 (r_x z_0 + h_x z(\overrightarrow{r}) - h_x z_0)}$$

$$(\mathrm{I}) = -\frac{h_x z(\overrightarrow{r})(f - z_0)^2}{z_0 f^2 (r_x z_0 + h_x z(\overrightarrow{r}) - h_x z_0)}$$

$$\therefore \frac{q_x}{\delta} = -p_x n \frac{h_x z(\overrightarrow{r})(f - z_0)^2}{z_0 f^2 (r_x z_0 + h_x z - h_x z_0)}$$

$$p_x = \frac{q_x}{n\delta} \frac{z_0 f^2 (z_0 - z(\overrightarrow{r}) - z_0 r_x / h_x)}{z(\overrightarrow{r})(f - z_0)^2}$$

$$p_x = \frac{q_x}{n\delta} \frac{z_0 f^2 (z_0 - z(\overrightarrow{r}))}{z(\overrightarrow{r})(f - z_0)^2} - \frac{q_x}{n\delta h_x} \frac{z_0^2 r_x f^2}{z(\overrightarrow{r})(f - z_0)^2}$$

From Equation (A.6) we know that:

$$\frac{q_x}{\delta h_x} = \frac{z_0 - f}{z_0 \mu}$$

Therefore

$$p_x = \frac{q_x}{n\delta} \frac{z_0 f^2 (z_0 - z(\overrightarrow{r}))}{z(\overrightarrow{r})(f - z_0)^2} - \frac{z_0 - f}{z_0 \mu n} \frac{z_0^2 r_x f^2}{z(\overrightarrow{r})(f - z_0)^2}$$

$$p_x = \frac{q_x}{n\delta} \frac{z_0 f^2 (z_0 - z(\overrightarrow{r}))}{z(\overrightarrow{r})(f - z_0)^2} - \frac{z_0 f^2}{z(\overrightarrow{r})(f - z_0)} \frac{r_x}{\mu n} \tag{A.10}$$

From the symmetry

$$p_y = \frac{q_x}{n\delta} \frac{z_0 f^2 (z_0 - z(\overrightarrow{r}))}{z(\overrightarrow{r})(f - z_0)^2} - \frac{z_0 f^2}{z(\overrightarrow{r})(f - z_0)} \frac{r_y}{\mu n} \tag{A.11}$$

A.2 Disparity

Let d_h and d_v be the horizontal and vertical disparity. The definition of disparity we adopt is this work is the shift we need to apply to a pixel of a given view to obtain its equivalent position on the view immediately on the right on the case of horizontal disparity and on the view immediately down for the vertical disparity.

From the definition of disparity, we obtain:

$$d_h = \frac{\Delta p_x}{\Delta q_x} = \frac{z_0 f^2}{n\delta (z_0 - f)^2} \frac{z_0 - z(\overrightarrow{r})}{z(\overrightarrow{r})} \tag{A.12}$$

$$d_v = \frac{\Delta p_y}{\Delta q_y} = \frac{z_0 f^2}{n\delta (z_0 - f)^2} \frac{z_0 - z(\overrightarrow{r})}{z(\overrightarrow{r})} \tag{A.13}$$

$$\frac{d_v}{d_h} = 1 \tag{A.14}$$

The value $\frac{z_0 f^2}{n\delta(z_0 - f)^2}$ is a constant that depends on the montage and setting of the camera.

From Equation (A.14) we can conclude that to estimate the horizontal disparity is equivalent to estimate de vertical disparity. We also observe that the objects falling on the focal plan of the camera ($z(\overrightarrow{r}) = z_0$) have a null disparity.

B Metadada files provided by Lytro camera

The Metadata is a file that provides information needed to convert the raw data into a coloured image, the identification of the device, and information about the settings of the camera for a given image. Below we have an example of the Metadata file on Lytro camera for a given photo.


```
{
    "type" : "lightField",
    "image" : {
        "width" : 3280,
        "height" : 3280,
        "orientation" : 1,
        "representation" : "rawPacked",
        "rawDetails" : {
            "pixelFormat" : {
                "rightShift" : 0,
                "black" : {
                    "r" : 168,
                    "gr" : 168,
                    "gb" : 168,
                    "b" : 168
                },
                "white" : {
                    "r" : 4095,
                    "gr" : 4095,
                    "gb" : 4095,
                    "b" : 4095
                }
            },
            "pixelPacking" : {
                "endianness" : "big",
                "bitsPerPixel" : 12
            },
            "mosaic" : {
                "tile" : "r,gr:gb,b",
                "upperLeftPixel" : "b"
            }
        },
        "color" : {
```

```
            "ccmRgbToSrgbArray" : [
                3.1115827560424805,
                -1.9393929243087769,
                -0.172189861536026,
                -0.3629055917263031,
                1.6408803462982178,
                -0.27797481417655945,
                0.078967012465000153,
                -1.1558042764663696,
                2.0768373012542725
            ],
            "gamma" : 0.41666001081466675,
            "applied" : { },
            "whiteBalanceGain" : {
                "r" : 1.046875,
                "gr" : 1,
                "gb" : 1,
                "b" : 1.24609375
            }
        },
        "modulationExposureBias" : -1.1520031690597534,
        "limitExposureBias" : 0
    },
    "devices" : {
        "clock" : {
            "zuluTime" : "2013-03-08T09:03:34.000Z"
        },
        "sensor" : {
            "bitsPerPixel" : 12,
            "mosaic" : {
                "tile" : "r,gr:gb,b",
                "upperLeftPixel" : "b"
            },
            "iso" : 95,
            "analogGain" : {
                "r" : 2.59375,
                "gr" : 1.90625,
                "gb" : 1.90625,
                "b" : 2.1875
            },
            "pixelPitch" : 1.3999999761581417e-006
        },
```

```
"lens" : {
    "infinityLambda" : 5.4248447418212891,
    "focalLength" : 0.013279999732971193,
    "zoomStep" : 590,
    "focusStep" : 993,
    "fNumber" : 2.0799999237060547,
    "temperature" : 25.4273681640625,
    "temperatureAdc" : 2855,
    "zoomStepperOffset" : 6,
    "focusStepperOffset" : 4,
    "exitPupilOffset" : {
        "z" : 0.061500057220458981
    }
},
"ndfilter" : {
    "exposureBias" : 4.046018123626709
},
"shutter" : {
    "mechanism" : "sensorOpenApertureClose",
    "frameExposureDuration" : 0.0040000001899898052,
    "pixelExposureDuration" : 0.0040000001899898052
},
"soc" : {
    "temperature" : 31.74169921875,
    "temperatureAdc" : 3178
},
"accelerometer" : {
    "sampleArray" : [
        {
            "x" : 0.027450980618596077,
            "y" : 0.97647058963775635,
            "z" : 0.364705890417099,
            "time" : 0
        }
    ]
},
"mla" : {
    "tiling" : "hexUniformRowMajor",
    "lensPitch" : 1.3999999999999998e-005,
    "rotation" : 0.0013527183327823877,
    "defectArray" : [],
    "config" : "com.lytro.mla.11",
```

```
            "scaleFactor" : {
                "x" : 1,
                "y" : 1.0002063512802124
            },
            "sensorOffset" : {
                "x" : -1.0148949623107912e-006,
                "y" : -3.69161456823349e-007,
                "z" : 2.5000000000000001e-005
            }
        }
    },
    "modes" : {
        "creative" : "auto",
        "manualControls" : "true",
        "exposureDurationMode" : "auto",
        "isoMode" : "auto",
        "ndFilterMode" : "auto",
        "exposureLock" : "false"
    },
    "camera" : {
        "make" : "Lytro, Inc.",
        "model" : "F01",
        "firmware" : "v1.0a138, vv1.1, Mon Oct 8 11:56:47 PDT 2012,
0c067f9721224660fcbb5cfc371a7243fa410ef1, mods=0, ofw=0"
    }
}
```