INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Application for Light Field Inpainting

Marta Almeida de Aragão Veiga Coelho

Master degree in Telecommunications and Computer Engineering

Supervisor:
PhD, Caroline Conti, Assistant Professor
Iscte-Instituto Universitário de Lisboa

Co-Supervisor:
PhD, Luís Eduardo de Pinho Ducla Soares, Associate Professor
Iscte-Instituto Universitário de Lisboa

November, 2022

TECNOLOGIAS
E ARQUITETURA

Department of Information Science and Technology

Application for Light Field Inpainting

Marta Almeida de Aragão Veiga Coelho

Master degree in Telecommunications and Computer Engineering

Supervisor:
PhD, Caroline Conti, Assistant Professor
Iscte-Instituto Universitário de Lisboa

Co-Supervisor:
PhD, Luís Eduardo de Pinho Ducla Soares, Associate Professor
Iscte-Instituto Universitário de Lisboa

November, 2022

For you Mum

Olívia Aragão

And for you Grandpa and Great Uncle

Mário Fernando

Adelino Jorge

Acknowledgements

In first place, I want to thank Iscte, for accepting me as a student, allowing me to learn and to get great experiences and opportunities. It has been an honour.

A big thank you to my supervisors, Prof. Caroline Conti and Prof. Luís Ducla Soares, for everything. For your contagious enthusiasm, for the good mood, for always wanting more and better, for the enormous patience you had to put up with me, for the encouragement, for all the constructive criticisms, and for the incredible work you did. Better supervisors than you are hard to find, and for that, I am forever grateful. I also thank Prof. Paulo Nunes from the Multimedia Signal Processing Group for Instituto de Telecomunicações (IT) for the fruitful discussions. A thank you to Instituto de Telecomunicações.

Another big thank you to Christine Guillemot, Mikaël Le Pendu, Pierre Allain, and Oriel Frigo, for providing and allowing me to work with their projects. I would not be able to do this project without your generous collaboration.

Thank you to my dad, for being my personal 'cheerleader'. For supporting me and believing in my dreams, for celebrating every victory and for helping me get up after every defeat. Thank you for teaching me the values of perseverance and commitment.

A special thank you to my dear aunts Teresa, Augusta, Margarida, Fernanda and Fátima, for taking care of me and teaching me to be kind, smart, considerate, humble, and joyful. An honourable mention to my grandma Maria Lucília, that is literally and figuratively my second mother. To my friend Teresa Belo.

To my lovey Pedro, not only for how impeccably you played the IT supporting role throughout this project, but also for all the emotional support, encouragement and care you provided me with.

Thank you to all my family and friends!

Resumo

Campos de luz é uma tecnologia multimédia que fornece uma experiência mais imersiva ao visualizar conteúdo multimédia com níveis mais altos de realismo, comparando a tecnologias convencionais de imagem. Esta tecnologia é promissora, principalmente para Realidade Virtual, pois exibe cenas capturadas do mundo real de forma que utilizadores as possam experimentar em todas as posições e ângulos, devido à sua representação em 4 dimensões. Por isso, esta é tecnologia em rápido crescimento, com tantos tópicos para explorar, sendo o inpainting o explorado nesta dissertação.

Inpainting de imagens é uma técnica de edição, permitindo sintetizar conteúdo alternativo para preencher lacunas numa imagem. Comumente usado para preencher partes que faltam numa cena e restaurar imagens danificadas, de forma que as modificações sejam corretas e visualmente realistas. É muito improvável que aplicar técnicas tradicionais de inpainting 2D diretamente a campos de luz resulte num inpainting consistente em todas as suas 4 dimensões. Normalmente, para fazer inpainting num conteúdo 4D de campos de luz, os algoritmos de inpainting 2D são usados para fazer inpainting de um ponto de vista específico e, seguidamente, os algoritmos de propagação de inpainting 4D propagam o resultado do inpainting para todos os dados do campo de luz 4D.

Com base nessa ideia de propagação de inpainting 4D, algumas técnicas foram recentemente propostas na literatura. Assim, esta dissertação propõe-se a conceber e implementar uma aplicação de inpainting de campos de luz que possa ser utilizada pelo público que pretenda trabalhar nesta área e/ou explorar e editar campos de luz.

Palavras-Chave: Campos de Luz, Inpainting, Propagação, Aplicação

Abstract

Light Field (LF) imaging is a multimedia technology that can provide more immersive experience when visualizing a multimedia content with higher levels of realism compared to conventional imaging technologies. This technology is mainly promising for Virtual Reality (VR) since it displays real-world scenes in a way that users can experience the captured scenes in every position and every angle, due to its 4-dimensional LF representation. For these reasons, LF is a fast-growing technology, with so many topics to explore, being the LF inpainting the one that was explored in this dissertation.

Image inpainting is an editing technique that allows synthesizing alternative content to fill in holes in an image. It is commonly used to fill missing parts in a scene and restore damaged images such that the modifications are correct and visually realistic. Applying traditional 2D inpainting techniques straightforwardly to LFs is very unlikely to result in a consistent inpainting in its all 4 dimensions. Usually, to inpaint a 4D LF content, 2D inpainting algorithms are used to inpaint a particular point of view and then 4D inpainting propagation algorithms propagate the inpainted result for the whole 4D LF data.

Based on this idea of 4D inpainting propagation, some 4D LF inpainting techniques have been recently proposed in the literature. Therefore, this dissertation proposes to design and implement an LF inpainting application that can be used by the public that desire to work in this field and/or explore and edit LFs.

Keywords: Light Field, Inpainting, Propagation, Application

Contents

List of Figures

List of Tables

Acronyms

2D - Two Dimensions

3D - Three Dimensions

3DTV - Three-Dimensional Television

4D - Four Dimensions

4K - Horizontal resolutions of around 4,000 pixels (the "K" stands for "kilo")

7D - Seven Dimensions

AR - Augmented Reality

CMake - Abbreviation of "Cross Platform Make"

DCT - Discrete Cosine Transform

DWT - Discrete Wavelet Transform

EPI - Epipolar Image

FoV - Field of View

GB - Gigabytes

GIMP - GNU Image Manipulation Program

GNU - GNU's Not Unix

GUI - Graphical User Interface

G'MIC - GREYC's Magic for Image Computing

GREYC - Groupe de Recherche en Informatique, Image et Instrumentation de Caen

HD - High Definition

HDR - High Dynamic Range

HMD - Head-Mounted Display

HVS - Trichomatic Human Vision System

LF - Light Field

MATLAB - MATrix LABoratory

MB - Megabytes

MI - Micro Image

MLA - Microlens Array

NHK - Nippon Hōsō Kyōkai Japan Broadcast Corporation

PDE - Partial Differential Equations

PS - Patch Sample

RGB - Red, Green, and Blue

ROI - Region of Interest

UHD - Ultra High Definition

UML - Unified Modeling Language

VR - Virtual Reality

Chapter 1
Introduction

In the last 50 years, there has been great progress in the development of multimedia devices, such as digital cameras, displays and smartphones, which can provide more immersive and closer to reality visual experiences to the end-users. In this context, computational imaging has made it possible for technology companies to build more advanced multimedia applications over the years. These new applications caught the attention of consumers, that can now create its own content and share it on social media. However, to build more advanced multimedia devices, applications, and services there are some desirable requirements to fulfil such as availability of higher bandwidth network, better resolution sensors and more computational power. Nowadays, the technical advances allow supporting 4K and Ultra High Definition (UHD) (Figure 1.1) spatial resolutions, higher temporal resolutions, and High Dynamic Range (HDR) contents.

Figure 1.1 - Difference between HD, UHD and 4K [1].

One available technology that did not have a good reception among end-users is Three Dimensional (3D) stereo visual technology [2]. This technology was created to provide more realistic and immersive visual experience to the public and it was used in 3D Television (3DTV) and 3D cinema markets. After the low public acceptance of this 3D visual technology, the sale and usage of devices that supported this technology decreased in the consumer market. The 3D visual technology failed in terms of sales because the public did not want to use the bizarre 3D glasses, they were not suitable for every person (for example, people that wear glasses and have other visual impairments), and the lack of parallax (the users could not move their heads because they would lose the visual experience). Technology companies that develop multimedia devices needed to build devices that support a better and acceptable visual technology. On that note, researchers had to develop new alternatives to immersive display technologies that would not fail in terms of public acceptance.

1.1. Context and Motivation

A potential imaging technology that provides new creative freedom and flexibility in many areas is Light Field (LF). This imaging technology enables a much richer description of the scene in every snapshot, including spatial and angular information. Therefore, it makes possible to achieve a more immersive experience with multimedia content that has higher levels of realism compared to other imaging technologies. In real-world scenes, LF can reproduce, with accuracy, stereo parallax, motion parallax, refractions, reflections, and volumetric effects. LF is a solution for Virtual Reality (VR), allowing acquisition and display of real-world scenes in a way that users can experience the captured scenes in every direction and every angle. As a result, the immersive experience becomes also more comfortable. LF imaging also provides a better sense of the materials within the scene by making it possible for the user to observe light reflections shift over the object's surface. As the high-quality consumer VR technology with positional head tracking provides a compelling way for viewing LF content, LF can stimulate and satisfy the sense of motion parallax. When the user moves his/her head in a scene, its depth and information about the materials, as well as lighting, are revealed in an engaging way. LF can also offer benefits over traditionally modelled synthetic 3D scenes, which requires a significant effort from the artist to build detailed models in order to allow photorealism. In contrast, LFs allows rendering real scenes by simply interpolating and extrapolating pixel data from the content that were previously captured. It has been shown that their rendering solutions

can be measured in milliseconds, independent of scene complexity, achieving the rates needed for compelling VR. For these reasons, LFs is an ideal photo and video representation for VR [2].

As the capture of LFs grows in popularity, the need for editing tools for this specific type of content is expected to rise as well. Inpainting techniques, which deal with the recovery of missing parts of an image, are a specific kind of image editing task that will be considered in this dissertation.

Image inpainting, also called image completion, is a process that aims to fill missing parts of an image and corresponds to an important task in computer vision. The goal is to restore damaged images, remove unwanted objects or retouch undesired regions of an image such that the modifications are correct and visually realistic. Recent 2D image inpainting solutions are using deep and machine learning techniques and are already able to achieve impressive results [3]. For example, the Magic Eraser feature on Google Pixel phone automatically detects objects in an image and the user can remove them with just a few simple steps, as illustrated in Figure 1.2.

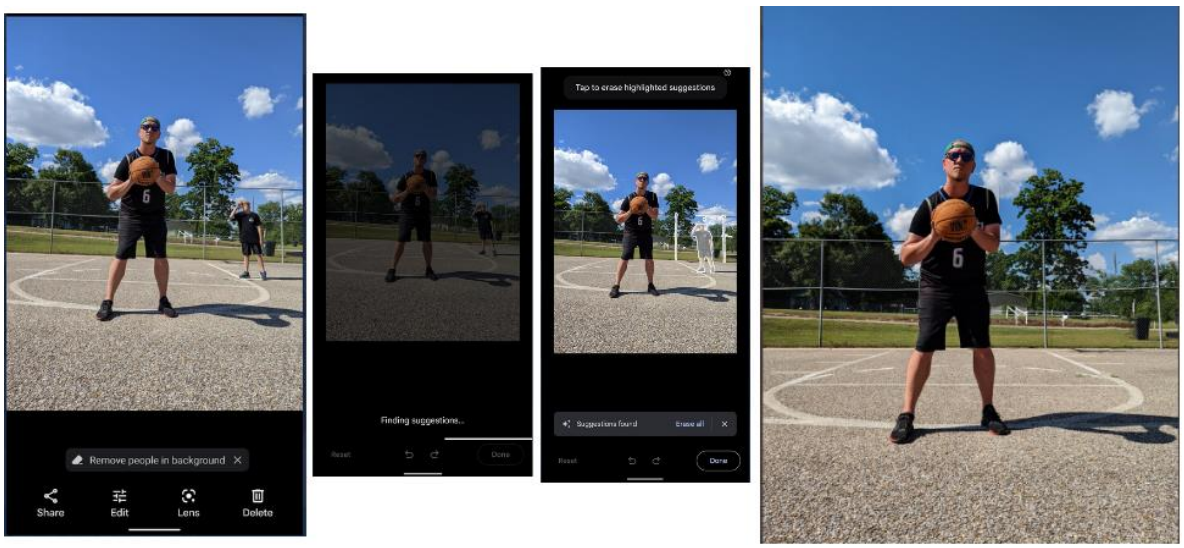


Figure 1.2 - How the Magic Eraser feature on Google Pixel phones works [4].

Different from 2D content, captured LF data typically consists of a 2D array of viewpoint images (also known as views or sub-aperture images) that are captured in a particular point of view, i.e., in every different 2D direction angle. An example of a LF capturing system is shown in Figure 1.3. In terms of inpainting, applying 2D inpainting techniques straightforwardly to LF images is very unlikely to produce reliable results. For instance, by applying the very advanced 2D deep learning inpainting algorithms independently to every viewpoint of a LF, an inconsistent inpainting result will most likely be achieved and a manual editing/refining step will have to follow. A better solution is to apply the inpainting technique to a single viewpoint and then propagate those results in a consistent fashion to the remaining viewpoints, while taking into account the relationship between viewpoints. For this reason, most of the LF inpainting techniques proposed in the literature rely on the propagation of the inpainting results across the various viewpoints of the entire LF consistently [5] [6] [7] [8] [9] [10].

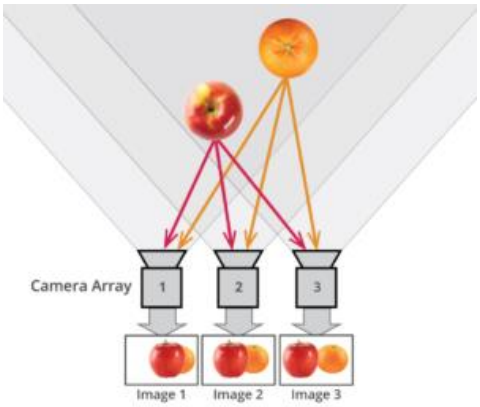


Figure 1.3 - Capturing a scene with a LF camera: different sensors (camera array) capture the same scene in different positions. The Image 1, Image 2 and Image 3 represent viewpoints [11].

Until today, there have been very few software tools dedicated to processing LFs. Some examples of these software applications are the Lytro desktop application, released by the camera manufacturer Lytro in 2012 [12], the Matlab Light Field Toolbox v0.4 library [13], and PlenoptiCam v1.0 [14]. However, the existing software applications have focused only on pre-processing tasks (e.g., demosaicing and calibration) and image rendering functionalities (e.g., perspective shift, refocus, and changing the depth-of-field) and, to the best of the author's knowledge, there has not been yet any software application that allows editing and inpainting LFs. Although some of the authors proposing LF inpainting algorithms have made their code publicly available, the program language adopted, and the configuration needed to execute the code.

1.2. Objectives

The main goal of this dissertation is to design and implement an editing application that allows the user to automate the task of inpainting LF content. To achieve this goal, the following main requirements have been identified:

1. Develop a free-of-charge and user-friendly LF editing application with Graphical User Interface (GUI) in order to facilitate the dissemination of knowledge in this field;
2. Integrate pre-processing features that could be useful for either researcher and students working in this field, or end-users. This would include the possibility, for instance, to facilitate the calibration of raw LF content;
3. Support a more scientific profile in the application by including features that could facilitate the experimentation and analysis of different LF inpainting algorithms proposed in the literature.

1.3. Methodology

The main stages in this work were:

1. Performing a bibliographic search to understand the state-of-the-art in LF processing and LF inpainting;
2. Studying the current existing tools and applications for LF pre-processing. This enables to decide which main features would be incorporated to the proposed application;
3. Designing, implementing, and testing the application for LF inpainting. This stage was performed in an iterative manner that converged to the final version. To develop the application, the MATLAB [15] scientific computing platform was used in a Linux environment;
4. Final testing of the developed application, which was used to assist in a comparison study between LF inpainting algorithms proposed in the literature.

1.4. Outline of the Dissertation

This document is organized in six different chapters, describing the different phases of the work until its conclusion.

The first chapter, which corresponds to the current one, introduces the dissertation topic, including the motivation to work in, the defined objectives, and methodology to achieve those objectives.

The second chapter presents a state-of-the-art review about LF imaging technology; it describes the technology, how it was developed, and how it is displayed and used.

The third chapter describes the 4D LF inpainting algorithms that exist in the literature. It starts by explaining the types of 2D inpainting algorithms that exist to inpaint a single 2D viewpoint. Then, it explains how the 4D inpainting propagation algorithms work, especially the different types of inpainting propagation from the 2D inpainted viewpoint to the whole LF data.

The fourth chapter introduces the developed application. It explains the goals, the tools and the datasets that were used, and its overall functionality.

The fifth chapter presents a comparative analysis between the 4D LF inpainting propagation algorithms that were implemented in the developed application, including their qualitative and quantitative evaluation.

The sixth and last chapter presents the work conclusions, and suggestions for future work directions.

Chapter 2
Light Field Imaging Technology: A Review

This chapter provides a brief review of the principles behind the LF imaging technology. The first section explains what a LF is, as well as its origins, with the LF being the basis for LF imaging technology. The following sections present a brief overview of the functional stages of LF imaging technology, which are essential do deliver LF content to end-users: acquisition, representation, rendering, application, editing, and display.

2.1. Light Field

LF imaging technology is based on the acquisition and description of information about light rays – the visible light information – to be able to display it later as accurately as possible. The light rays carry a lot of visual information about the 3D environment, being able to do things that were not possible before. The term LF was first defined by Gershun in 1936 [16]. It is a model that describes the distribution of light rays in free space, with Gershun describing the concept of LF as being "light beams that propagate in a straight line through a homogeneous medium and which is the carrier of radiant energy in a space". This model was later improved by Bergen in 1991 [17] [18].

The Bergen model is the plenoptic function and describes light in a scene as a function of position, angle, wavelength and time. Rays in space can be parameterized by coordinates x, y and z, angles θ and φ, wavelength λ and time t. This is a seven (7D) dimensional function which describes the light rays intensity at every possible location in space with the coordinates x, y and z, travelling toward every possible direction specified by the angles θ and φ , with the range of wavelengths λ and at any time t. The plenoptic function shown in Equation (1) is illustrated in Figure 2.1.

$$P(x, y, z, \theta, \varphi, \lambda, t) \tag{1}$$

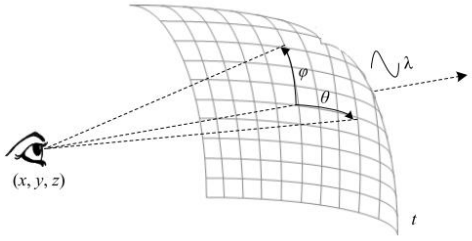


Figure 2.1 - The 7D plenoptic function.

Obtaining the plenoptic function for a given scene with its seven dimensions can be very challenging. The challenge resides in the fact that to sample the plenoptic function for a scene in a 7D representation requires a lot of data. To appropriately sample the data, Levoy and Hanrahan proposed in [19] three assumptions to reduce the dimensionality of the plenoptic function to only four (4D) dimensions:

1. *Static scene* – If the scene is static, the time variable *t* becomes fixed and the plenoptic function can be then reduced to P($x,y,z,\theta,\varphi,\lambda$).
2. *Constant Radiance along its Path (Free-Space)* – With this assumption, the light rays are continuous lines that travel through space without changing their properties. Therefore, the radiance of any light ray in space can be always obtained by tracing it back to a selected surface, which means that the plenoptic function can thus be reduced to P($x,y,\theta,\varphi,\lambda$).
3. *Trichromatic Human Vision System (HVS)* – In the human eye, each type of the three photosensitive cells (cones) in the retina has its maximum sensitivity in a different wavelength, corresponding roughly to the primary colours Red (R), Green (G) and Blue (B). Since (most) visible colours can be represented as a combination of R, G and B, it is possible to reduce the wavelength dimension by assuming three different plenoptic functions (one for each R, G and B components). This way, for each colour component, a 4D plenoptic function is defined as P(x,y,θ,φ).

In addition to these simplifications, the 4D plenoptic function can also be represented in a different way, as defined by Levoy and Hanrahan in [19]. This alternative representation uses a two-plane parameterization to represent the 4D function in Cartesian coordinates. In this parameterization, a specific light ray intersects the first plane at coordinates (x,y), that define the spatial location of the ray, and it is propagated in free-space until it intersects the second plane at coordinates (u,v), which indirectly define the propagation direction of the ray. This function was baptized as the 4D LF (or Lumigraph).

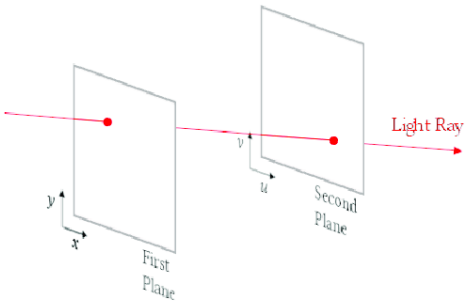


Figure 2.2 - Two-plane parameterization for the 4D plenoptic function [20].

$$L\,(x, y, u, v) \tag{2}$$

Light can be seen as a scalar radiance (i.e., one value for each component R, G and B) traveling along straight lines (rays) with different propagation directions. This 4D LF function will then be used to represent a LF image.

The organizational framework for the rest of the Chapter 2 is as illustrated in Figure 2.3.

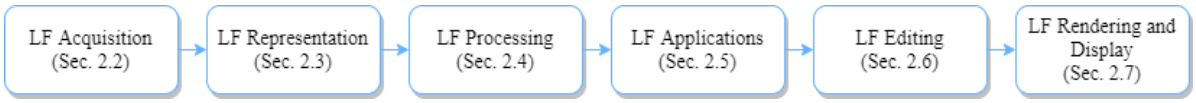


Figure 2.3 - Organizational framework of Chapter 2.

2.2. LF Acquisition

The LF acquisition represents the capturing part of the data (scenes) in 4D LF. For that purpose, this section focuses on existing devices or methods for LF images acquisition.

A 2D conventional camera captures a 2D projection of a LF on its sensor plane, by integrating the light rays that hit each pixel from every direction. On the other hand, a LF device/camera can capture the angular and spatial information of light rays, but the device sensors can only measure the information from two dimensions of a scene at a single moment, because every sensor is located in different positions inside the camera, capturing the scenes in different directions and angles. In conventional cameras, the angular information is not captured because the light rays are integrated in the spatial position of the sensors. The acquisition for devices that support LF, in case of 4D LF, needs to capture multiple samples along the angular dimensions.

In LF cameras, the focus plane, depth-of-field and viewpoint are not fixed at the moment of the capture, the image can be adjusted while rendering in post processing.

Existing LF acquisition approaches can be divided into three fundamental categories:

- *Light field capture with multi-camera array:* This approach uses a multi-camera array to build a new 3D volumetric representation of the 3D space. A number of viewpoints with parallax are captured using an array of multiple cameras in a matrix arrangement, as shown in Figure 2.4. In context of parallax, when viewpoints have parallax in both directions it is called full parallax. The viewpoints captured with this setup have full parallax.

Figure 2.4 - Multi-camera array from [21].

To obtain the spatial and angular densities, the following approach is used: the acquisition of spatial density is done at each camera sensor and depends on its resolution, while the angular density is obtained through the distance between neighbouring cameras (i.e., baseline) measured in millimetres to centimetres (providing more angularly dense LF). An example of a camera is the Stanford multi-camera array

proposed in [21] and shown in Figure 2.4. This camera is composed of one hundred custom cameras and supports reconfigurable arrangement of the array for three different scenarios: panoramic video with HDR, synthetic aperture photography, and widely spaced 3D scenes. This setup gives provides high resolution data in the spatial, angular, and temporal dimensions.

- *Light field capture with camera gantry:* In this category, a time-sequential capture approach is used: a single image sensor is used to capture multiple samples of the LF through multiple exposures. For this purpose, it uses a sensor mounted on a mechanical gantry to acquire de LF at distinct positions, at different viewpoints and at different instants of time. The spatial density depends on the camera's sensor resolution, while the angular density depends on the images position. The FoV (Field of View) depends on the degrees of freedom that is supported by the gantry structure. An example of cameras that use such configurations is the Stanford LF gantry (Figure 2.5).

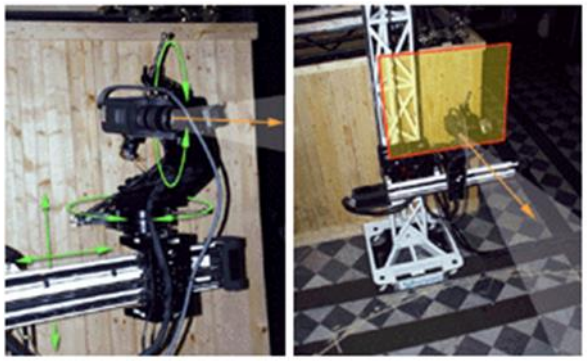


Figure 2.5 - Stanford camera gantry [22].

This camera has four degrees of freedom in terms of movements: translation along x and y axes and rotation along θ and ϕ axes. Even though this category of cameras provides some flexibility and lower cost, they are restricted for capturing static scenes.

- *Light field capture with lenslet camera:* The cameras with this technology adopt the concept proposed by Lippman [23], also known as integral, holoscopic or plenoptic imaging. Using this concept, LF content with full parallax can be acquired by using a single-tier sensor camera overlaid with a Microlens Array (MLA) (Figure 2.6).

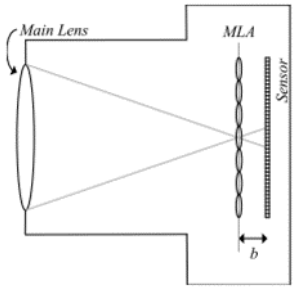


Figure 2.6 - Lenslet camera [18].

Also referred to as lenslet array, the MLA can be seen as a tiny 2D array of cameras with a very small baseline, sampling the 4D LF and organizing it in a conventional 2D image, known as the lenslet image, which corresponds to the raw LF image. There are several cameras that uses this configuration, with the two more relevant ones being the Raytrix [24] and the Lytro [25].

A LF camera essentially comprises a main lens and an image sensor overlaid with an MLA. The light rays coming from the main lens will be thus focused on the sensor (Figure 2.6). All the light rays coming from a given depth plane that are converging to the MLA plane will intersect at the MLA and then diverge until they reach the image sensor.

2.3. LF Representation

The representation of LF raw data, acquired from the previous stage, must be convenient and the viewpoints consistent. After acquisition, the representation can have one of the following (raw) formats, given a certain set of devices requirements: lenslet representation and full parallax multiview representation. For specific requirements, different kinds of representation should be used.

2.3.1. Lenslet Representation

For a LF acquired using a lenslet camera, the LF content is represented as a 2D grid of microlens images (micro-images, elemental images, and macro-pixels) as shown in Figure 2.7 (a).

For a lenslet representation there is no need for conversion and/or processing of the raw acquisition when a lenslet camera is used. In this case, the LF data is represented as a 2D image comprising a grid of micro-images as shown in (Figure 2.7 (a)).

2.3.2. Full Parallax Multiview Representation

For LF acquired using a multi-camera array or a camera gantry, the LF content is represented by a 2D grid of viewpoints and, usually, each viewpoint has the same spatial resolution, as shown in Figure 2.7 (b).

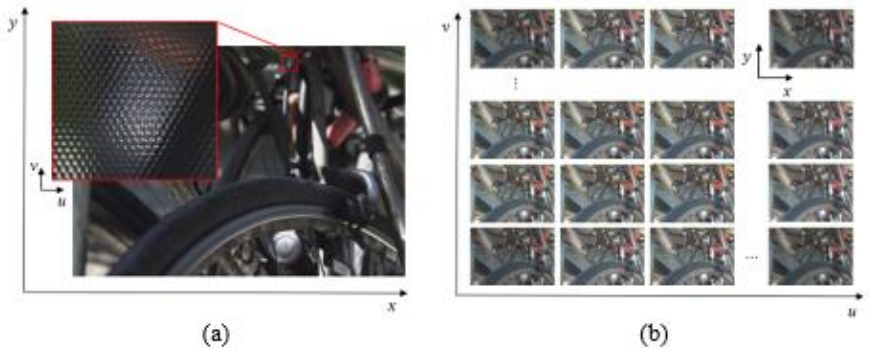


Figure 2.7 - Lenslet format (a), and full parallax multiview format (b) [18].

In the case of LF captured using a camera array, the acquisition representation corresponds to the full parallax multiview representation (Figure 2.7 (b)). Differently, the LF content captured using a LF camera needs to be pre-processed and converted.

2.3.3. EPI images

As discussed in section 2.1, a 4D LF image consists of an array of viewpoints of the captured scene, with varying angular coordinates u and v. An Epipolar-Plane Image (EPI) represents a 2D slice of the 4D LF. An EPI image is a decomposition of the light ray distribution in an LF image. EPI-based representation can help estimate the depth/disparity of the objects from the slope of the built EPI (see Figure 2.8 (a)). In Figure 2.8 (b), an EPI built by stacking together viewpoints in the same column of the 4D LF array is shown, with each column corresponding

to the angular coordinate u. In each viewpoint, a slice is taken in a particular horizontal plane by fixing the directional coordinate x.

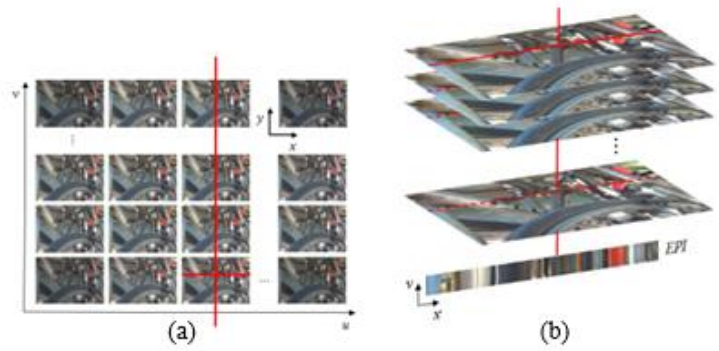


Figure 2.8 - Extracting an EPI from the 4D LF from [18].

2.3.4. Focal Stack

The focal stack is a collection of images with different focus settings. Each image is focused on a different depth. The collection of images used to represent the focal stack can be exhibited in a specific depth. Several cameras do not provide sufficient control over the depth setting because some have a fixed aperture, while others have an aperture that can be too large or too small to produce the required amount of blur. To solve this problem, focal stack images can be used because the amount of depth can be achieved. It is a good solution because most cameras have focus controls, allowing the capture of a stack of images focused on different distances [26].

2.4. LF Processing

The introduction of Lytro and Raytrix cameras to the market made it possible for LF images to be used for numerous applications. As a result, the demand for algorithms to process 4D LF also increased. Some of these algorithms presented at recent academic conferences work on depth estimation, super-resolution, and compression.

The LF data is high of dimension, causing a problem with storage and communication, and it needs high processing power and the angular-spatial resolution trade-off inherent to LF capture devices. This high volume of data (containing spatial and angular information of the scene) will be able to provide many of post-capture processing capabilities, such as re-focusing, different viewpoint rendering and depth estimation, from a single exposure [27].

The four-dimensional LF representation enables the depth map estimation of a scene. Data collected from a LF capture can provide different depth values of the scene, making the depth map estimation possible and more robust and precise. To make real-time depth estimation in LF it is required small computing times and a lot of resources because it is necessary to process high dimensional data. Constraints and cues are used to estimate the depth map from a LF image, because they take advantage of all the viewpoints together. This first depth estimation cannot avoid noise, occlusions or inherent matching uncertainty caused by texture-less regions. An approach can be later used – depth refinement. This estimation smoothens out the outliers and produces the final depth map. Figure 2.9 shows the initial (Figure 2.9 (b)) and final (Figure 2.9 (c)) depth map from an input LF.

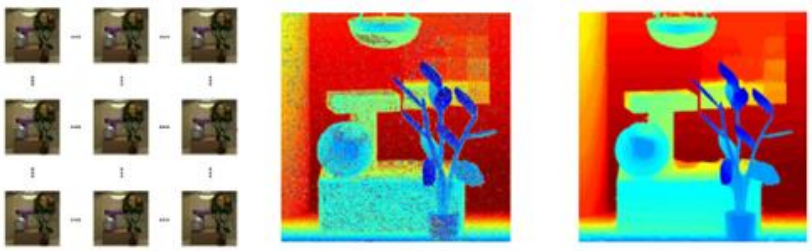


Figure 2.9 - Typical depth estimation pipeline using LF data from [17].

2.5. LF Applications

LF imaging technology has applications that were developed as the computational power increased over the years, and LFs have grown in popularity [17].

2.5.1. Synthetic Aperture Imaging

This application allows to “see through” an occluder, by synthesizing a sufficiently large aperture. The part of the image that has the object occluded and the occluder are at different parallaxes and depth layers, so each viewpoint can capture a small portion of the object. Levoy et al. [28] used this application to reconstruct objects that were partially occluded. The goal is to reconstruct in a single viewpoint the occluded object, in [28] it was an object occluded by plants. A single camera, with an appropriate focal plane, was used to capture all viewpoints reflected by an array of planar mirrors. The result was a virtual viewpoint with a wide aperture synthesized, with the object in focus and the occluders were blurred out. The result can be seen in Figure 2.10.

Figure 2.10 - LF acquired by capturing an array of planar mirrors. (a) are the viewpoints of the captured LF; (b) is the synthetic aperture image [28].

A camera array system can provide clear viewpoints of objects using synthetic aperture techniques, instead of tracking the objects in a passive way like the conventional single-camera tracking algorithms.

2.5.2. Future of Surgery

In 2016, Raytrix developed the first autonomous robot surgery Raytrix 3D cameras. This camera and software can position the robot in hard to access places and in critical situations. In dental imaging, Raytrix also developed a handheld dental 3D oral-scanner that has superior 3D

acquisition of Raytrix cameras; one of its features is that it is good for visualizing difficult-to-access objects avoiding occlusions (Figure 2.11). Other applications are: flow mechanics (particle tracking), human body and skin scanning, face detection/recognition, plant analysis & pharmacy, among others [29].

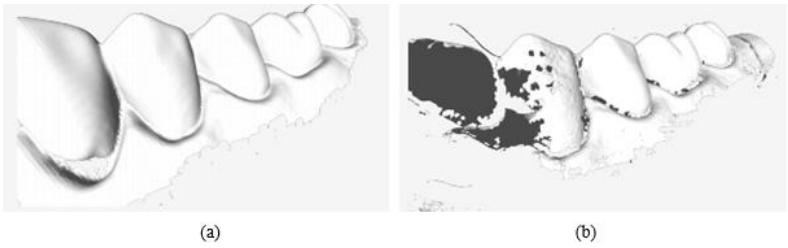


Figure 2.11 - Comparison between new LF camera (a) and standard oral-scanner (b).

2.6. LF Editing

LF content contains complex directionally varying effects – reflections, refractive objects, or occlusions. The edits on LF can be done locally or globally, if the edits are made in a precise place in the image, or on the full LF (the edits must be propagated to all viewpoints on the LF).

In editing 2D images, there are GUI that allow the user to edit a 2D image. LF editing can be a challenge because most of the currently available editing software are designed for two-dimensional content. LF data is redundant; any local edit on a LF viewpoint needs to be propagated consistently to preserve this redundancy. For a software to apply a user interface with LF data, the LF properties must be considered to present the visual information in an accurate way, and to preserve the amount of redundancy.

- Editing interaction*:* This editing method consists in the way that the users edit LF. The position of the edit consists of the angular-spatial position, and to specify the position in depth within the LF focus or parallax are used. In Figure 2.12, an example edit can be seen, using standard 2D displays allowing LF editing by gesture tracking [17].

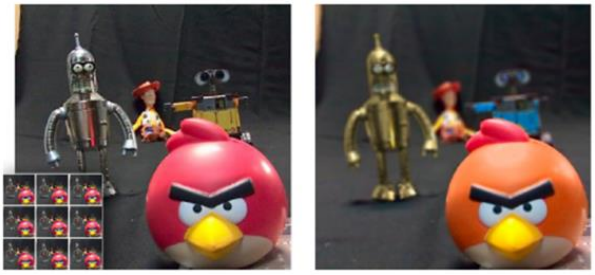


Figure 2.12 - LF captured with the Lytro camera [17].

Another editing application for LF images is inpainting. The inpainting allows the user to fill missing parts in a scene and restore damaged images such that the modifications are correct and visually realistic. The inpainting application is discussed on Chapter 3.

2.7. LF Rendering and Display

2.7.1. LF Rendering

A LF can be interpreted as a 2D collection of 2D images of a scene, thus creating a 4D array of pixels.

To render light rays the two planes' schemes can be applied. First, their intersection coordinates with the two planes are computed. Then, the nearest sixteen sampled rays are used to interpolate the virtual ray. In Figure 2.13 is shown the two planes representation, these planes represent the 4D LF, in which four of the sampled rays are applied for interpolation. The above idea is called LF rendering [17].

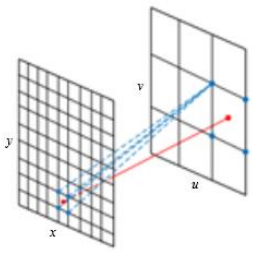


Figure 2.13 - Rendering a virtual viewpoint using the recorded LF: For a virtual ray, the nearest 16 sampled rays are used to perform an interpolation [9].

2.7.2. LF Display

In the last two decades, there have been created different technologies capable of displaying LF. A LF display adds the ability to see the scene from different viewpoints, this ability does not exist in conventional 2D or 3D displays. The perception of depth is shown not only through binocular disparity, but also through motion parallax, due to the different viewpoints. In this concept, there is no need to use glasses or other hardware, and it is displayed different viewpoints of the LF to both eyes and for each viewpoint within the FoV of the display. The recent development of sensor and optical manufacturing provide a more natural and immersive experience in display technologies [17].

The display technologies that are currently available for LF content visualization are:

1. 2D Displays - Displays a 2D version of the LF content. The content must be rendered from the decoded LF content.
2. Multiview Autostereoscopic Displays - Glassless display technology. This technology allows creating a more natural 3D illusion to the end-user, presenting a unique perspective as the user moves horizontally around the display. Multiple viewpoints need to be rendered from the LF content and delivered to the display.
3. AR and VR Displays - These technologies display different perspectives as the user moves through the scene, visualizing the LF content with full parallax. Some AR and VR solutions have proposed to take advantage of a microlens based, or a mirror-based LF imaging technology for creating a more natural visualization in AR and VR HMDs.
4. LF Displays - Technology that uses an optical setup similar to the one used in lenslet cameras. It can be also designed for LF visualization, as proposed by Nippon Hōsō Kyōkai (NHK) Japan Broadcast Corporation. Another LF display technology uses a very dense number of viewpoints to create a replica of the 4D LF, proposed by Holografika, Ostendo, and Looking Glass Factory [20].

Chapter 3
Light Field Inpainting: Literature Review

As explained previously in section 1.1, images can be edited through a technique called image inpainting. Inpainting is a technique that allows making edits on images, such as restoring damaged images or filling holes in a scene. The goal is for the resulting image from the inpainting process (particularly the target area) to be filled in a way that it is not perceptible to the viewer that the image was edited, and that the modifications are correctly made. In the inpainting process, a target area is selected/delimited and the 2D inpainting algorithm treats that target area as if it was a hole in the image, and its objective is to fill that hole in a coherent way so that it is not noticeable that the final image has been edited. An example of the inpainting technique is illustrated in Figure 3.1.

Figure 3.1 - Example of inpainting: (a) original image, (b) target area (in this case the target area is the person), (c) inpainting result [30].

In this chapter, the various available techniques for inpainting LF content are organized and briefly described. The inpainting in 2D images can be performed in a target viewpoint (usually the central view). Then, the inpainting result is propagated to all viewpoints in the 4D LF image, by the 4D LF inpainting propagation algorithms. 2D inpainting methods (not specific for LF) can be sub-divided in two main classes: diffusion-based methods and patch-based methods. The inpainting propagation of 4D LF content can be done with 2D-to-4D propagation techniques or with integrated 4D inpainting method. An overview of the inpainting methods is illustrated in Figure 3.2.

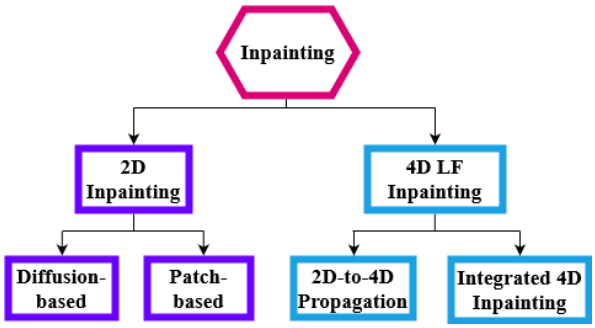


Figure 3.2 - Organization of inpainting methods for LF inpainting and propagating.

3.1. 2D Inpainting

In terms of 2D inpainting, the inpainting is performed in 2D images, and the result is also a 2D image with the target area inpainted. Many different solutions have been proposed in the past and can be used for 2D-to-4D propagation. Two 2D inpainting methods are: diffusion-based and patch-based.

3.1.1. Diffusion-based method

The diffusion-based method [31] propagates information from surrounding regions to the inpainted areas and can efficiently recover thin holes, basically by diffusing neighbouring pixel information. A problem with this method is that the local image statistics may have changed after inpainting process, such as local variance difference and noise patterns in the inpainted image, making it easy for the forensic algorithms and user to locate the inpainted regions. In Figure 3.3 there is an example of inpainting done by diffusion-based method. The area to be inpainted is filled with black pixels **Ω** and the inpainting result is in Figure 3.3 (b) surrounded by the dashed curve. The inpainting process restored the missing pixel information in an image with neighbouring pixel information.

Figure 3.3 - Diffusion-based method: the target area is illustrated on (a), inpainting result on (b). The inpainting was done considering the colour information of the surrounding pixels [31].

3.1.2. Patch-based method

This method, also described in [31], fills in missing regions by getting information from similar regions and is more suited for object removal thanks to its ability to fill larger holes by using patches of texture taken from the known regions of the image, as illustrated in Figure 3.4. With the patches taken from the known region, it is possible to have the highest possible patch similarity level. The patch-based method assumes that the information in the missing part of the picture resides elsewhere in the picture. A problem with this method is that it requires more processing power because it consistently involves searching and comparing. An inpainted image by this method is illustrated in Figure 3.5.

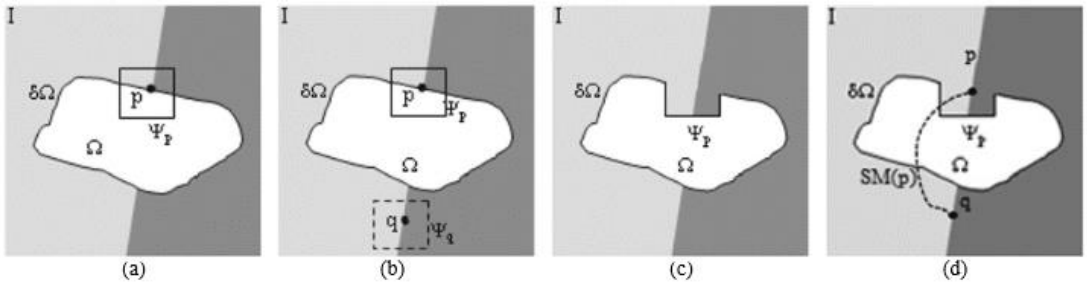


Figure 3.4 - Patch-based method: (a) Priority estimation, (b) Patch match, (c) Inpainting, (d) Offset definition. The region to remove is p and the patch q is the one that has higher similar level; hence it is the one that fills the p region [32].

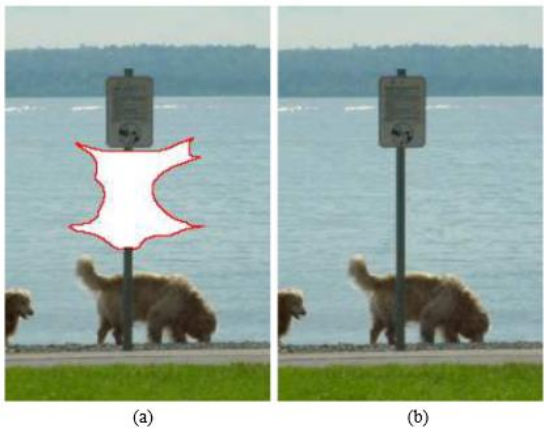


Figure 3.5 - Patch-based inpaint method: The target region is illustrated on (a), inpainting result on (b) [33].

3.1.3. Application of 2D inpainting techniques to 4D LF

In case of inpainting, plenty of algorithms already exist in the 2D case. To apply independently these methods to each viewpoint of an LF could not result in a consistent inpainting in the angular direction. As illustrated in Figure 3.6, the two images were inpainted independently, and the results are not consistent with each other (when changing the angle perspective, the rock should not disappear). For example, in an image with large holes in a background with complex textures and geometry that need to be recovered, advanced patch-based methods are not remarkably effective, giving unsatisfactory results.

Therefore, to have a satisfactory process of inpainting of LF images, a manual editing is necessary, but manually editing each viewpoint image in a consistent manner is extremely tedious [9].

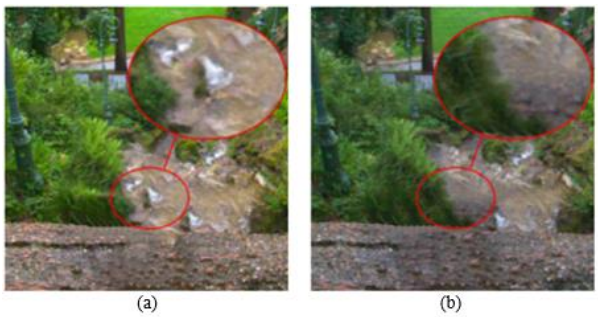


Figure 3.6 - Inpainting propagation through viewpoint images. On (a) there is a rock, but when changing the angle perspective, the rock suddenly disappears (b) [5].

One and the most difficult challenges in LF inpainting is the high volume of data to be processed. For user interaction and LF editing - segmentation, object removal, inpainting, colorization - now common with 2D images editing is made difficult due to this challenge. In fact, LF image editing is still poorly developed in terms of practicality and user-friendliness. Another challenge - besides computing complexity - is the fact that the edits have to be consistent among viewpoints, i.e., the target area in the inpainting result image should look similar in all viewpoints. When visualizing the full 4D LF, the user should be able to visualize all objects in the LF image; this scenario does not happen in Figure 3.6, once the user changes perspective, the rock disappears.

3.2. 2D-to-4D Inpainting Propagation

This section describes the LF 4D propagation techniques that rely on first inpainting a 2D viewpoint of the LF content (normally the central view) and then propagate the inpainting result to the rest of the LF content. The 2D-to-4D inpainting propagation techniques described here are the ones present in the literature, organized in Figure 3.7.

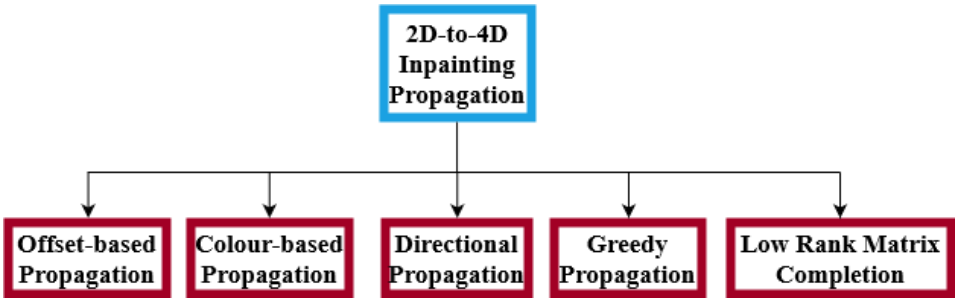


Figure 3.7 - 2D-to-4D propagation techniques that are currently present in the literature.

The general steps to perform a 2D-to-4D inpainting propagation are:

1. *2D viewpoint inpainting* – The first step consists in inpainting a 2D viewpoint of the LF content (usually the central view) with a 2D inpainting algorithm.
2. *Propagation to the full 4D LF image*– After the considered 2D viewpoint been inpainted, the second step is to propagate the resulting inpainting to all the viewpoints in the 4D LF image.

Since the used 2D inpainting methods can essentially be any (with some exceptions) of the available techniques in the literature for 2D images, the focus of this section will be mainly on the techniques used for propagating the inpainting to the 4D LF image. A total of five different categories of techniques for propagating the inpainting made in the 2D viewpoint to the entire 4D LF image exist (Figure 3.7) and are described in the sections below. For the developed application, two techniques for propagation will be used: directional propagation [8] [9], and low rank matrix completion [5]. The common main challenge these techniques face is to make a coherent and consistent propagation of the inpainting texture to all viewpoints on the LF.

3.2.1. LF inpainting based on Offset Propagation

In this technique, proposed in [7], the central view is inpainted using 2D patch-based method. Between the filled patches are offsets, which together with their best match in the known regions, are propagated to the entire LF. Figure 3.8 illustrates how it works. Offset is the position of the patch. For each pixel that is edited in the central view, it looks at the location of source patch (chosen from the 2D patch-based method), then finds the corresponding source location in the other viewpoints.

This process is done independently in the views. It propagates patch offsets to ensure that viewpoints are edited using pixel colours sampled from the image itself, and the edits applied in viewpoints are consistent with the ones in the central view. To keep consistency between viewpoints, it must do the offset relative to the disparity. When changing between viewpoints, the perspective changes in the same way in all viewpoints. Nonetheless, this technique has the disadvantage of high complexity and/or from angular view coherency. In [7], the authors use offset-based propagation (instead of colour-based propagation) because it generates more natural appearance variations that are consistent with the original data.

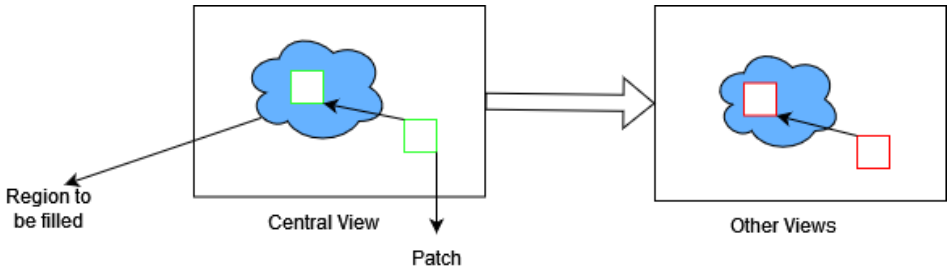


Figure 3.8 - Overview of the offset-based Propagation method.

3.2.2. LF inpainting based on Colour-Based Propagation

The goal of this technique, also in [7], is to find the regions that were removed from the central view to other viewpoints and fill them. An overview of the technique is illustrated in Figure 3.9. The way that this technique works is the following: it finds the corresponding pixel location in the central view of a pixel that needs to be synthesized in one viewpoint. Then, it finds the topmost layer at this location that is visible by examining the layer masks, and it copies the pixel value on this layer. This method can create satisfactory results, but all corresponding pixels across different viewpoints will have exactly the same synthesized colour.

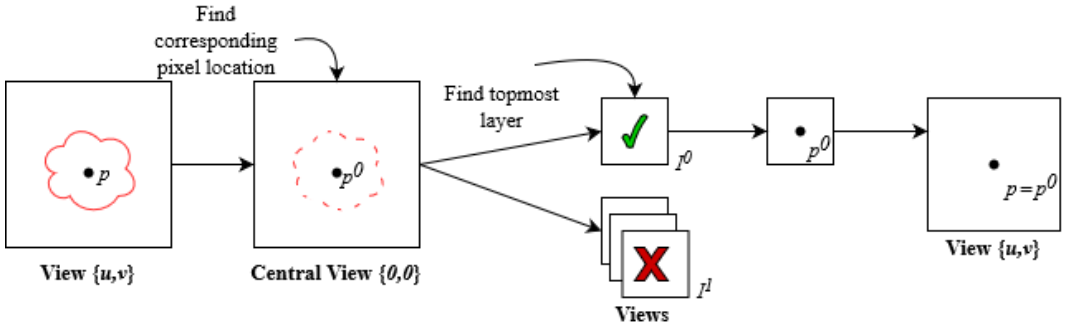


Figure 3.9 - Overview of the colour-based propagation method: a pixel p in the region to be removed in a viewpoint of the LF. The method finds the corresponding pixel location on the central view, which is p^0. In this location, it finds the topmost layer that is visible by examining the layer masks. In this flowchart, the topmost layer is layer I^0. In this layer, the method copies the pixel value p^0 to p.

3.2.3. LF inpainting based on Directional Propagation

The directional propagation is a propagation technique where the inpainting is propagated in a directional way – in the *u,v* directions. First, the central view is inpainted with a 2D inpaint algorithm, then the edits are propagated in the angular directions *(u,v)*, see Figure 3.10. To avoid inconsistencies between views, it can be used disparity information.

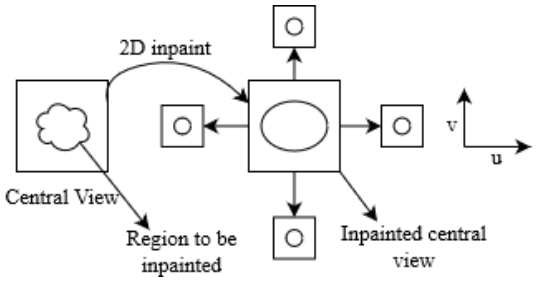


Figure 3.10 - Diagram of the directional propagation method.

1) Angular Forward Warping

This propagation method [8] uses the computed disparity in the masked area to propagate the inpainted area of one viewpoint to all the other viewpoints of the LF using angular forward warping. An overview of this method is illustrated in Figure 3.11. The propagating information wraps directly in directions *(u,v)*, and requires the scene geometry (disparity information inside the mask) in the inpainted region. This information does not exist in the LF scene, because inpainting a viewpoint can modify depth geometry in the inpainted area. It is assumed that depth is homogeneous in local regions having the same colour. Two main advantages of this method are: there is no need to repeat the whole pipeline of disparity computation and inpaint propagation for each EPI stripe, and direct warping is very fast as compared to diffusion.

Despite the disparity field is much noisier, this noise does not affect the inpainted disparity values as it turns out to be automatically removed via the colour-guided interpolation, reducing computational complexity while improving disparity coherency in the masked region. The inpainted disparity allows performing a simple angular warping to propagate inpainted texture across the whole LF. A disadvantage is that sometimes the final result can have ghosting and cracking effects, because it can have a local divergence of disparity and multiple pixels with different depths and colours converge to the same pixel in the warped image.

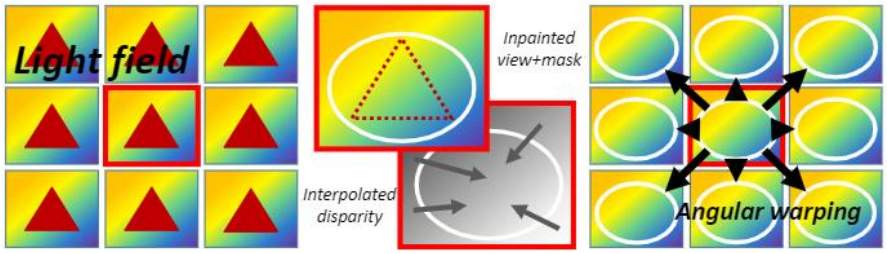


Figure 3.11 - Overview of the angular forward warping propagation method [8].

2) Epipolar Plane Diffusion

This propagation method [9] exploits smoothness in the EPIs to coherently propagate information across views using PDE-based diffusion. An overview of this method is illustrated in Figure 3.12. In the first step of this method, the central view is recoloured and inpainted manually with a 2D image manipulation tool. Then, it propagates the inpainted region from the central view to the other viewpoints. The epipolar plane diffusion allows to propagate an editing from the central view of a LF row (or column), to the other viewpoints of same row (or column) - *u*,*v*. Yet, an interpolation per EPI may lead to spatial inconsistencies in the disparity maps. This method also assumes that depth is homogeneous in local regions having the same colour. The LF viewpoint to inpaint in 2D has to be the central view.

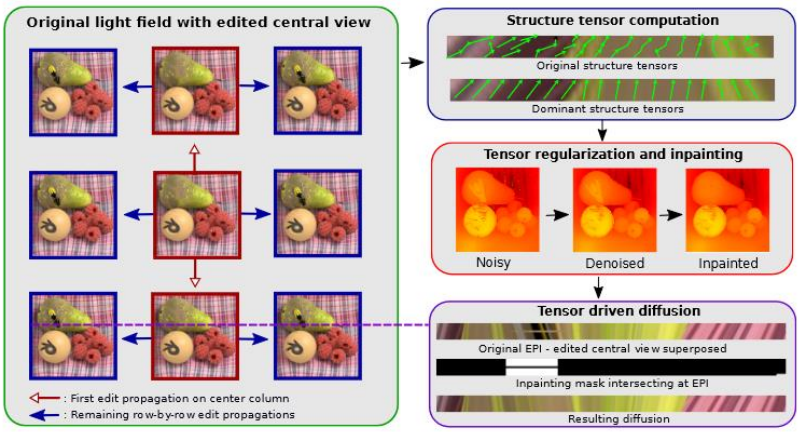


Figure 3.12 - Overview of the epipolar plane diffusion propagation method [9].

3.2.4. LF inpainting based on Greedy Propagation

In this propagation technique, the inpainting is propagated to the entire LF in a greedy fashion. The algorithm does not pay attention to features such as disparities, angular coherency, and pixel value. The LF viewpoint to inpaint does not have to be the central view. The viewpoints in a LF are usually awfully close. If the viewpoints are not close, the results of this propagation method are not satisfactory. The region to be removed in the LF is propagated to all viewpoints of a LF, and then it is removed in every viewpoint, as illustrated in Figure 3.13.

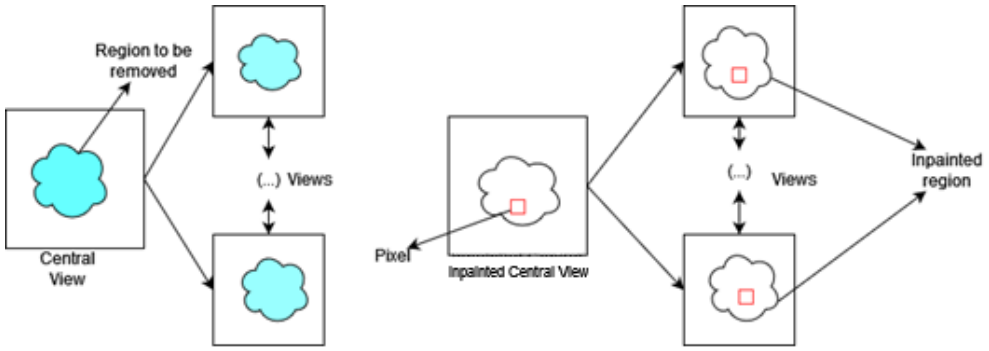


Figure 3.13 - Overview of the greedy propagation method. The region to be removed is propagated to all the other viewpoints, in the same position. Then, the inpainting in each viewpoint is done by looking for the best matching patches (corresponding pixel value) in the inpainted central view.

1) Bi-Directional Similarity

This propagation method [10] uses EPI images to reveal the depth structure of the scene. The goal is to create a novel 4D LF patch consistency measure for avoiding synthesis of inconsistent patches into the edited LF viewpoints. The central view is inpainted using 2D patch-based method. An illustration of this method in an image can be seen in Figure 3.14. During the propagation and search phase, it is considered 4D neighbours rather than 2D neighbours. The method automatically transfers the specified region to other viewpoints. Then, it removes the target specified regions from all viewpoints and fills in the holes due to content removal. This method moves between viewpoints (patch per patch) in a greedy way, the global consistency of the LF is not guaranteed, causing high computational complexity and angular view incoherency.

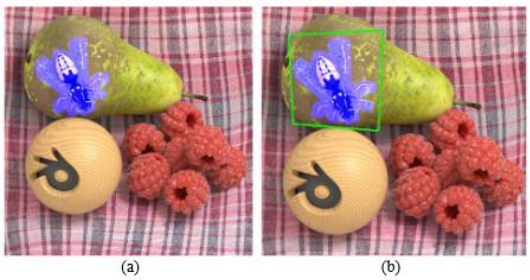


Figure 3.14 - Illustrates the region to transfer. The user specifies a region in blue to be removed in one view and this region is automatically transferred to other views. (b) shows the transferred region in one of the other views, and the green quadrangle indicates the transferred bounding box [10].

2) Depth Invariant Propagation

In this propagation method [6], the first step is to inpaint the central view using 2D patch-based method. The other viewpoints are inpainted by searching for the best matching patches in the inpainted central view. An illustration of an example of this method using re-colorization can be seen in Figure 3.15. The second step is to find the corresponding pixel in the central view for each pixel in LF viewpoints. The authors measure the intensity differences to find the corresponding pixels, and after the corresponding pixels are found, they utilize the edit value of the corresponding pixel as the edit value of a pixel in the LF image. However, this method assumes that the region to inpaint has a similar value with neighbouring visible regions, and nearby pixels with similar colour should have similar result. This method does not require any accurate depth information, and the viewpoints are processed separately. For these reasons, this is a greedy propagation method, resulting in angular inconsistencies.

Figure 3.15 - Pipeline of the depth invariant propagation method (re-colorization example) [6].

3.2.5. LF inpainting based on Low Rank Matrix Completion

In this propagation technique [5], the purpose is to make a consistent propagation of the inpainted region to all the other viewpoints of the LF. It is based on matrix completion, but assuming a low rank prior for the LF data. The first step is to inpaint the central view, either manually or using a 2D inpaint algorithm. Then, it performs a method for randomly generating homography warpings. This method builds wrapped version of the inpainted central view, with the rows *(x,y)* and the new pixel values (area of the image that was inpainted). With the homography warpings that were generated previously, this method builds a Low Rank Matrix – *M*, see Figure 3.16. This method guarantees consistency between views.

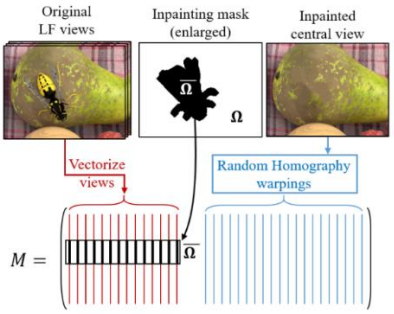


Figure 3.16 - Overview of the low rank matrix completion technique. A matrix is formed by vectorising each viewpoint and concatenating the resulting column vectors. A row of the matrix contains the pixels' values in all viewpoints at a fixed (*x*,*y*). Then, there are generated several additional viewpoints by warping the inpainted central view with a set of random homography projections [5].

3.3. Integrated 4D Inpainting

The integrated 4D inpainting does the inpainting of the 4D LF at the same time. In this method, a 4D segmentation technique [34] can be used to detect automatically objects in the 4D LF. The user can select the object(s) to be inpainted from the scene (the user can visualize the result of the automatic segmentation to adjust different viewpoints). Then, the inpainting process is done in every viewpoint simultaneously, as a consistency metric between the various views is maximized, as illustrated in Figure 3.17.

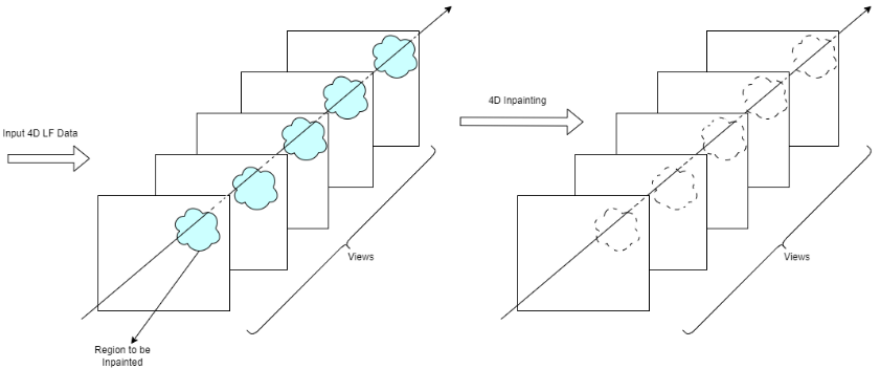


Figure 3.17 - Overview of the Integrated 4D Inpainting.

The pixels position, value and colour are outlined directly in every view.

With this method it is possible to achieve consistency in the entire 4D LF data. As far as the author's knowledge there are currently no solutions of this type available in the literature.

Chapter 4
Developed LF Inpainting Application

The application hereby developed aims to be a friendly and simple application for users (students, researchers, teachers or even enthusiasts) to obtain a full 4D inpainted LF images.

Currently in the literature, the project/code from some 4D LF inpainting propagation algorithms are not available. The various 4D LF inpainting propagation algorithms can be found in academic papers (which includes the algorithm description, its architecture, and inpainting results), but some of them come with no project/code attached to it. The 4D inpainting propagation algorithms that exist currently in the literature are:

- Light field inpainting propagation via low rank matrix completion [5];
- Spatio-angular consistent editing framework for 4D light field images [6];
- PlenoPatch: patch-based plenoptic image manipulation [7];
- Fast light field inpainting propagation using angular warping and colour-guided disparity interpolation [8];
- Epipolar plane diffusion: an efficient approach for light field editing [9];
- Light field image editing by 4D patch synthesis [10].

From these algorithms, the ones that were applied to the application were [5], [9] and [8].

Their source-code was available, and the project owners kindly provided the source-code to the author. In addition, these three algorithms are the most recent and popular ones in the LF inpainting propagation topic.

There is a great list of articles and documentation about the LF imaging technology and editing/inpainting techniques. One can learn a lot about the topic from these documents. The same cannot be said about testing and working with LF inpainting propagation algorithms. A theoretical description of such algorithms was done in Chapter 2, but from a practical standpoint, it is difficult to find resources to work with them.

This application is designed as a ‘centralized’ resource of LF inpainting propagation algorithms, where anyone interested in the topic can test and work with the three inpainting propagation algorithms, and even add more 2D inpainting algorithms and 4D inpainting propagation algorithms. Therefore, the task of learning and understanding the LF imaging and inpainting can be achieved, not only by reading articles/documentation, but also by testing and working on it.

This application allows the user to import LF images (through PlenopticCam [14]), select the target viewpoint to inpaint (create a mask and inpaint with two 2D inpainting algorithms), and propagate the inpainting to the whole LF image with three 4D inpainting propagation algorithms. The following image (Figure 4.1) is a UML diagram of the developed application.

There are four main modules:

- Import data module;
- Mask of the target view module;
- 2D inpaint of the target view module;
- Generate inpainted LF with 4D inpainting propagation algorithms module, with the following algorithms (Figure 4.1):
    - LFIPLRMC: Algorithm Light Field Inpainting Propagation via Low Rank Matrix Completion [5].
    - EPD: Algorithm Epipolar Plane Diffusion [9].
    - FLFIPAWCGDI: Algorithm Fast Light Field Inpainting Propagation using Angular Warping and Colour-Guided Disparity Interpolation [8].

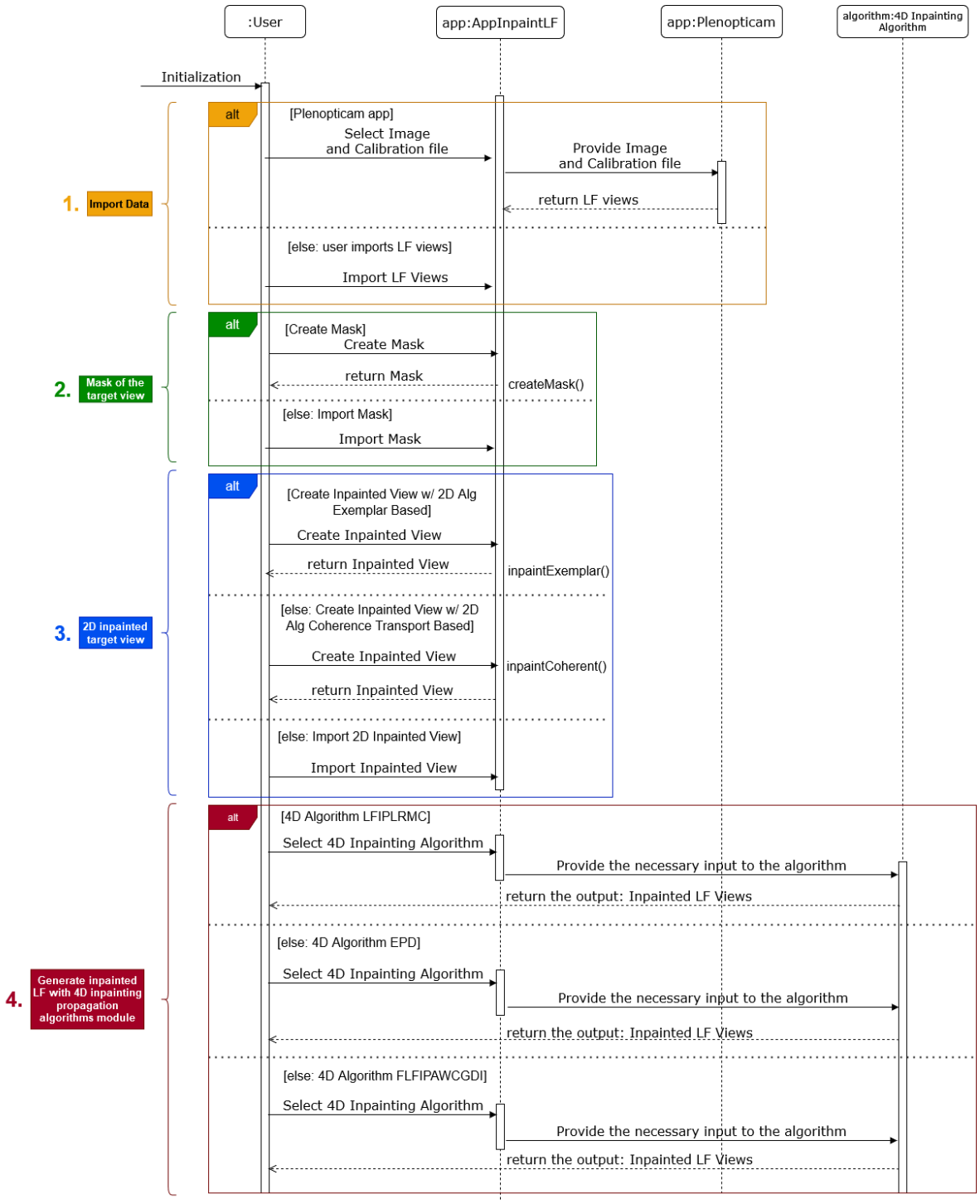


Figure 4.1 - UML diagram of the developed application.

In the import data module, if the user does not have a LF image (all the viewpoints of a LF image), he/she can use the application to import data. The PlenoptiCam [14] application is integrated onto the developed application, and it generates the full LF viewpoints for the user. The user has to provide a LF photograph (e.g., lfr format) and a calibration source data (archive or respective raw file from Lytro Illum). The GUI of this module is represented in Figure 4.4.

In the mask of the target view module, the developed application allows the user to select the target viewpoint to create a mask to be used for 2D inpainting. If the user already has a mask, he/she can import it into the application.

In the 2D inpainted target view module, the application allows the user to inpaint the target viewpoint, with two types of 2D inpainting algorithms (exemplar-based image inpainting algorithm, and coherence transport-based inpainting algorithm). If the user already has an inpainted target viewpoint, he /she can import it into the application.

If the user already has its own LF image, mask and 2D inpainted target viewpoint, the application allows him/her to import them and use as the input for the three 4D inpainting propagation algorithms.

Once the right input is inserted onto the application, the user can select one of the three 4D inpainting propagation algorithms (one at a time), and the application will run the source-code of the selected algorithm and provide its output (which is the full inpainted 4D LF image).

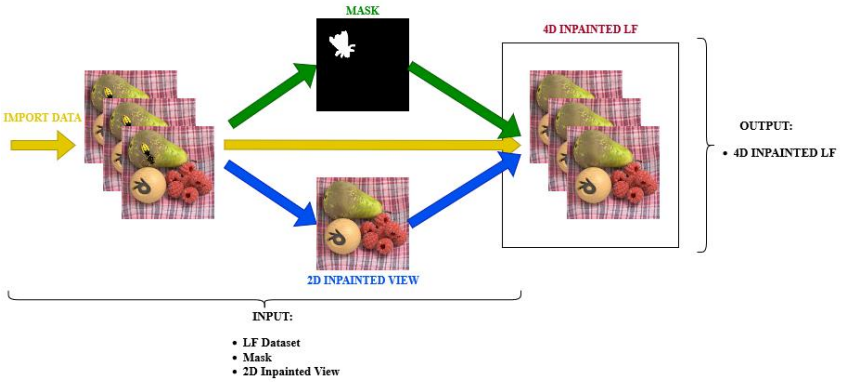


Figure 4.2 - Overview of the developed application.

Figure 4.2 illustrates the overview of the developed application. The first step is to import the LF image. The three 4D inpainting propagation algorithms require the following inputs: at least one mask and one 2D inpainting of the target viewpoint (normally the central view). With these input elements, the 4D inpainting propagation algorithms generate a new LF image, with all the viewpoints inpainted (2D-to-4D propagation).

4.1. Import data

The 4D LF images used in this application came from:

- Computational light field imaging project INRIA [35] [36];
- A synthetic dataset with 24 synthetic, densely sampled 4D LF with highly accurate disparity ground truth, depth and disparity maps for 81 viewpoints of the additional scenes [37] [38];
- A synthetic LF image '*Still Life'* provided along with the code from [9][1];
- The LF image 'Totoro Waterfall' was decoded with the D. Dansereau's Light Field Toolbox [13], on Figure A.1 and Figure A.2.

On Figure A.1and Figure A.2, the LF image was taken from a first generation Lytro camera, the viewpoints were extracted with the toolbox of Dansereau et al. [13]. On Figure A.3, Figure A.4, Figure A.5, and Figure A.6 the LF images are from a scene taken by a Lytro Illum camera and the viewpoints were extracted with the PlenoptiCam application [5] [35]. The LF images on Figure A.7 and Figure A.8 are from a synthetic LF, and the viewpoints are already available [38].

The real-life LF images used this application were acquired with:

- A first generation Lytro camera [39] [36]
- A Lytro Illum camera [5] [35]

To import all viewpoints of a LF image the user needs an lfr or lfp file format (that contains the captured scene) and a calibration file (tar format) for camera calibration data of the captured scene relative to the Lytro camera.

The raw LF data can be decoded with D. Dansereau's Light Field Toolbox [13] or using Lytro's desktop application [12].

With the datasets that have the image file (lfr or lfp) and the calibration file (tar), the viewpoints can be extracted with the help of the PlenoptiCam application [14]. This application is user friendly; it can be used as a desktop app and on command line (Windows and Linux). It is cross-platform, open-source software for scientific LF computation with few dependencies and a lean graphical user interface. It uses the raw exposures from a plenoptic camera and can calibrate an image taken by a plenoptic camera and extract sub-aperture images (viewpoints).

[1] Project code and '*Still Life*' LF image was provided by Oriel Frigo, the author of the algorithm epipolar plane diffusion.

This application supports custom-types of plenoptic cameras and is thus not limited to Lytro's image data. The user can just import into the application the raw image and calibration file, and the PlenoptiCam application can provide the following output (see Figure 4.3):

- sub-aperture images (viewpoints) and view animation given as gif;
- refocused images and refocus animation given as gif;
- raw image file given as tiff;

Figure 4.3 - A duck sitting in the grass. Two viewpoints, refocused images, and a raw image file.

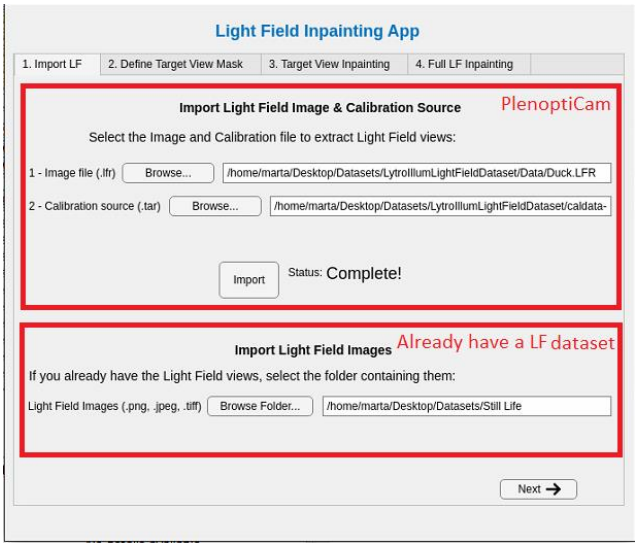


Figure 4.4 - Import data can be done with the help from the PlenoptiCam app, integrated in the developed application.

4.2. Mask

To inpaint a LF, the first task to do is to create a mask, where the user selects the central view (normally) and draws, using the pointer of the mouse device, the region-of-interest (ROI) on the selected viewpoint. This is performed by a MATLAB function called *drawassisted* [40].

The drawassisted function creates an *AssistedFreehand* object that allows the user to draw interactively a ROI, the contours of the object in the image with the mouse, or programmatically by using name-value arguments. To draw the ROI, the user must position the pointer on the image, click and draw the contour of the object to be removed. Then upon release, the whole region will be all covered up with blue. By double-clicking with the pointer inside the ROI, the shape will close, and the ROI will be finished. If the ROI region did not catch certain areas of interest, the user can right-click on the line and select 'Add Waypoint'. A little dot will appear, and the user can move it as he/she pleases, until the desired area is covered by the ROI.

MATLAB also allows the user to zoom in, zoom out and grab the image, for a better and more precise drawing. This process is illustrated in Figure 4.5.

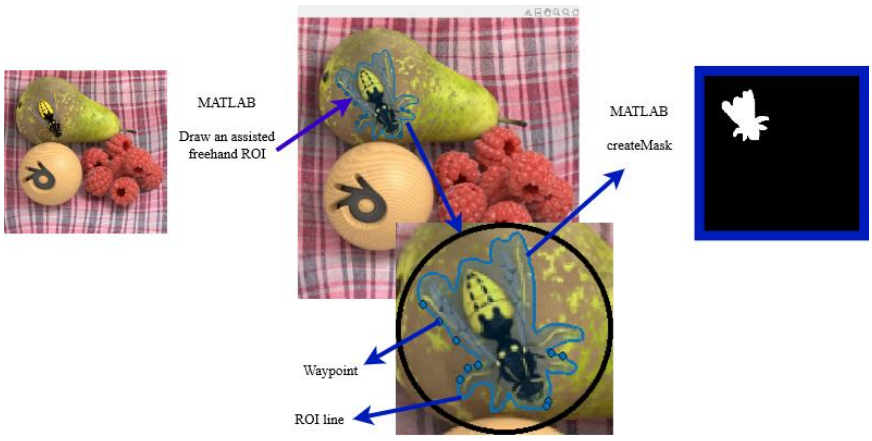


Figure 4.5 - Creation of the mask from a target viewpoint using *drawassisted* and *createMask* functions from MATLAB.

From an inpainting perspective, the region of a viewpoint that the user wants to inpaint (normally the central view) is inpainted in the region of the white pixels in the mask. The local inpainting occurs inside a mask, on the region defined by the user with the *createMask* function (white pixel region).

The 4D inpainting propagation algorithms take as input the mask and uses it to help inpaint the target area. When the user finishes creating the mask, the application displays the selected target viewpoint and the result of the mask creation, as seen in Figure 4.6.

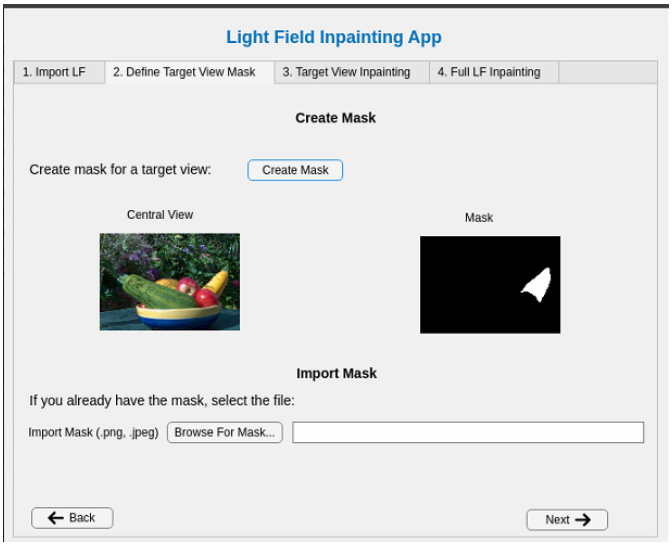


Figure 4.6 - In the developed application, the user selects 'Create Mask' and proceeds with it. When the user finishes creating the mask, the selected target viewpoint and the mask created are displayed on the application window.

4.3. 2D Inpainting Algorithms

Once the mask is created, it is necessary to inpaint the target viewpoint. In this application, is it provided two 2D inpainting algorithms:

- Exemplar-based image inpainting algorithm [41] [33], and
- Coherence transport-based inpainting algorithm [42] [43]

The user can also import an inpainted target viewpoint into the application (inpainting previously done with a different software/2D algorithm). The GUI of this module is represented in Figure 4.7.

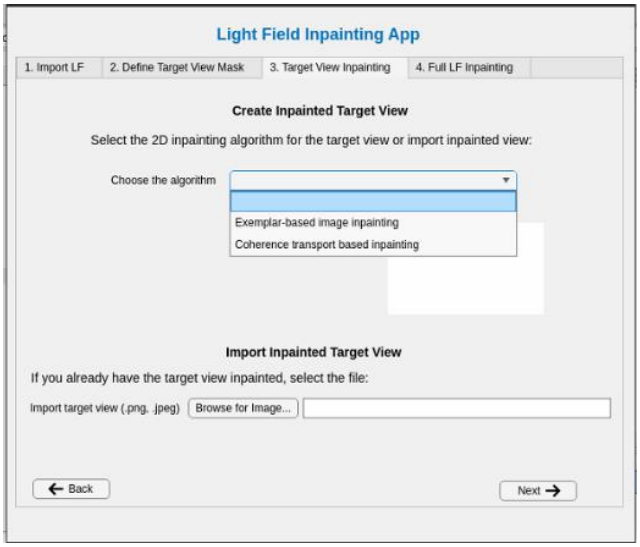


Figure 4.7 - 2D inpainting of the target view. There are two 2D algorithm options.

4.3.1. Exemplar-based image inpainting algorithm

This 2D algorithm is a patch-based method that restores target regions in the input image. It identifies the target region from the input image, thanks to the mask. For every patch centered on a boundary pixel in the target area, it computes the patch priority by using the gradient or tensor method (fill order). Given the target patch, it searches for the best-matching patch in the source region and copies it to the target patch. It updates the input image, mask, and patch priority value (patch size), and finally repeats previous steps until the target regions are inpainted (inpainted viewpoint with this algorithm is illustrated in Figure 4.8) [41].

Figure 4.8 - Filling the bee region from the image using the exemplar-based algorithm, with the central view (a), the target area in white with red boundary on (b), and the inpainting result on (c).

4.3.2. Coherence transport-based inpainting algorithm

This 2D algorithm is a diffusion and pixel-based method for removing objects and filling regions in images. The inpainting task starts from the boundary pixels of the target area. The inpainting value for a pixel is estimated from its coherent neighbouring pixels with known values (inpainted viewpoint with this algorithm is illustrated in Figure 4.9).

The inpainting value for a pixel in the target area is the weighted average of known pixel values within its inpainting radius. The known pixels along the coherence direction are assigned higher weight value than the incoherent neighbouring pixels (smoothing factor) [43].

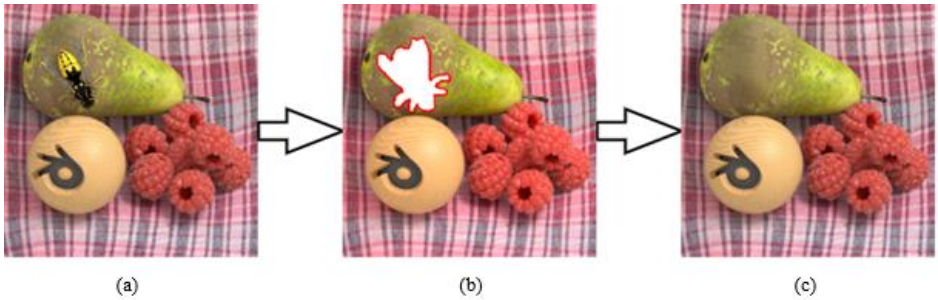


Figure 4.9 - Filling the bee region from the image using the coherence transport-based algorithm, with the central view (a), the target area in white with red boundary on (b), and the inpainting result on (c).

On the developed application, user can select which 2D inpainting algorithm to inpaint the target viewpoint (Figure 4.7). The application displays the window illustrated in Figure 4.10. The application displays the result of the 2D inpainting on the target viewpoint. The user can change the 2D algorithm parameters to have different inpainting results. For the exemplar-based image inpainting algorithm, the parameters are the fill order and patch size. For the coherence transport-based inpainting algorithm, the parameters are the smoothing factor and radius.

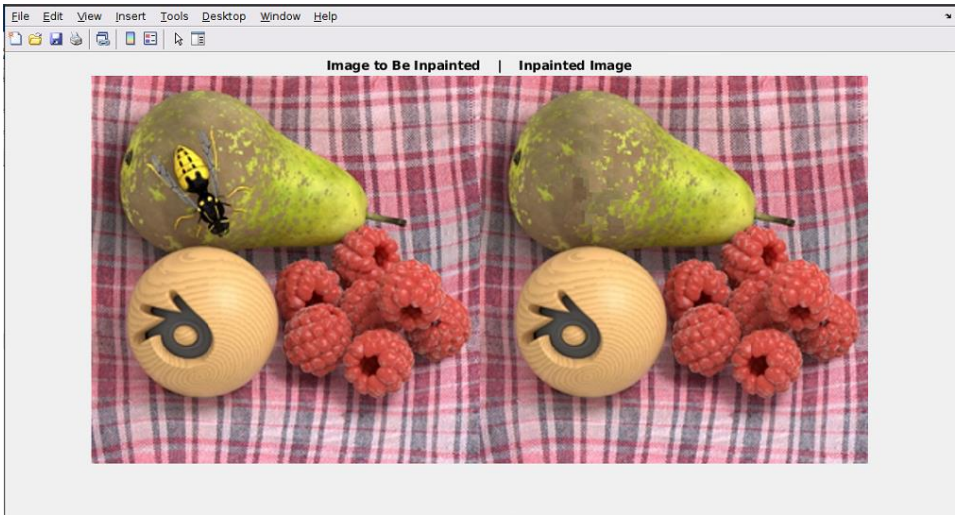


Figure 4.10 - The application will display the 2D inpainting result. The user can optionally change the algorithms parameters. At the end, the user can save the inpainted image.

4.4. 2D-to-4D Inpainting Propagation

The three 4D inpainting propagation algorithm used in this application are described in section 3.2. The three algorithms are:

- *Low Rank Matrix* - Light field inpainting propagation via low rank matrix completion [5].
- *Epipolar Plane Diffusion* - Epipolar plane diffusion [9].
- *Angular Warping* - Fast light field inpainting propagation using angular warping and colour-guided disparity interpolation [8].

The application saves the path from where the LF images (viewpoints) are, mask and inpainted target viewpoint. When user selects ‘Inpaint’ in each algorithm, the selected algorithm gets the inputs elements from the application and runs. The output from each algorithm is the entire 4D LF image (all viewpoints) inpainted. The output is stored in a path of the user’s choice, as illustrated in Figure 4.11.

□

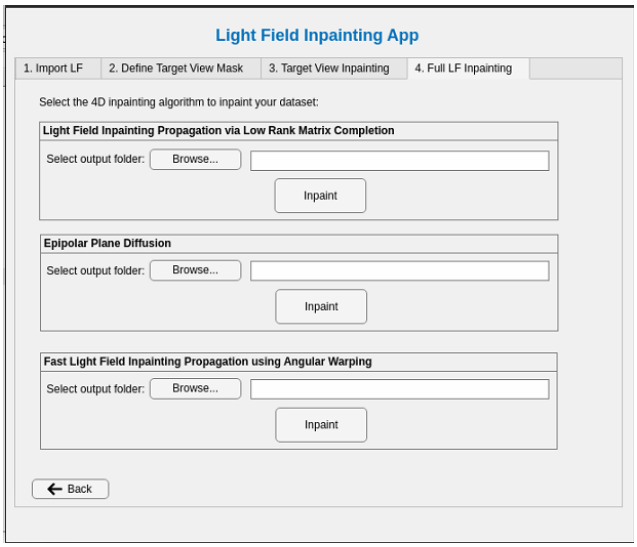


Figure 4.11 - Final window of the developed application: user can choose between three 4D inpainting algorithms to generate the full 4D LF inpainted views.

Chapter 5
Performance Evaluation

This section presents experiments with 2D inpainting algorithms and 4D inpainting propagation algorithms, which are present in application developed in this paper.

5.1. Test Conditions

The inpainting of the central view is performed by the two 2D inpainting algorithms (exemplar-based image inpainting algorithm for patch-based method and coherence transport-based inpainting algorithm for diffusion-based method). For the experiments, it was necessary to inpaint some central views manually with [44], because patch-based and diffusion-based methods did not provide a satisfactory inpainting result. The mask that is used is the same for the same LF image, i.e., the two 2D inpainting methods use the same mask to inpaint the central view. The test and results were applied for the inpainting of the synthetic LF "*Still Life*", "*Bedroom*", "*Bicycle*", "*Herbs*", and for real scenes captured with Lytro Illum Camera "*Bee_2*", "*Duck*", "*Fruits*", "*Toys*" and one captured with Lytro First Generation Camera "*Totoro Waterfall*".

The 4D inpainting propagation algorithms used in this application are:

- Light field inpainting propagation via low rank matrix completion [5]. Built with MATLAB and C++.
- Epipolar plane diffusion [9]. Built with Python, CMake and C++.
- Fast light field inpainting propagation using angular warping and colour-guided disparity interpolation [8]. Built with C++ and CMake.

5.2. Central View 2D Inpainting Results

The central view inpainting results with the 2D inpainting algorithms are shown on Figure 5.1, Figure 5.2, Figure A.9, Figure A.10, Figure A.11, and Figure A.12.

In each figure is illustrated the central view and mask used for inpainting the 2D central view. The 2D inpainting algorithm used are the exemplar-based image inpainting algorithm, the coherence transport-based inpainting algorithm, and manually inpainted performed by [44].

The exemplar-based image inpainting algorithm is a type of patch-based method and the coherence transport-based inpainting algorithm is a type of diffusion-based method, so from Figure 5.1 until the last figure they will be referenced as patch-based and diffusion-based, respectively.

The images that were inpainted with the patch-based and diffusion-based method presented a good inpainting result on the target area, manually inpaint was not necessary (see Figure 5.1, Figure A.10, and Figure A.12). The images Figure 5.2, Figure A.9, and Figure A.11 were inpainted with the patch-based, diffusion-based and manually because the inpainting with the patch-based and diffusion-based method did not produce satisfactory inpainting results in the target area, hence it was necessary do perform a manually inpaint with [44]. This technique is a simple-to-use software and allows non-specialist public to use it, through a GUI. It can be used on open-source GIMP2 software with a dedicated filter to the G'MIC plugin.

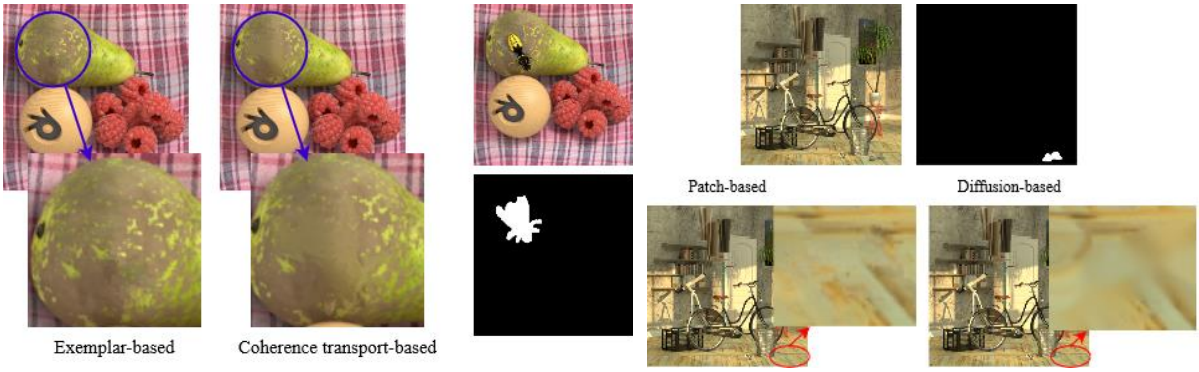


Figure 5.1 - The two inpainted central views and the mask used for the tests on "*Still Life*" (left) and "*Bicycle*" (right), with two different inpainting methods.

Figure 5.2 - Inpainting of the central view of the image "*Totoro Waterfall*" (left) and "*Herbs*" (right), with three different inpainting methods.

The 2D inpainting methods work differently with each image. On "*Bee_2*" image, on Figure A.9, the best inpainting result was achieved by manually inpainting the central view, but the patch-based method inpainting result comes in second place because the colour change on the leaf is not subtle. The diffusion-based could not inpaint evenly the separation between the leaves and the leaf shadow. On "*Toys*" image, Figure A.11, the same situation occurs with manually inpainting providing better visual results than the two tested 2D inpainting algorithms, and on the patch-based it can be seen a lot of scattered patches. In addition to this case, the diffusion-based inpainting technique did not achieve acceptable results in the target area.

For the "*Duck*" image, shown in Figure A.10, the best results have been achieved with the patch-based method, since the results obtained with the diffusion-based method in the target area look out of focus, even if the target area is small. This is due to the texture mix in the target area.

In Figure 5.1, on "*Totoro Waterfall*" image, three different inpaintings were performed in the totoro target area: the manually inpainting provided the best result and is the only one considered acceptable, due to the different depth layers. As such, the images obtained with the other inpainting methods are not going to be used for the propagation of this LF image.

The images of "*Fruit V1*" and "*Fruit V2*" (Figure A.12) have different target areas. As seen in the mask in the first image, it targets the yellow fruit, while the other mask targets the stem of the peach. In the first image "*Fruit V1*", the target area was inpainted better with the patch-based method, this because the size of the target area is big. In the second image "*Fruit V2*", the target region was inpainted better with the diffusion-based method, due to the size of the target area being small.

For the image of "*Bicycle*", Figure 5.1, the tested 2D inpainting methods performed well.

The "*Herbs*" image (Figure 5.2) presents not satisfactory results with the patch-based and diffusion-based methods; this is due to the texture mix of the neighbour region. Therefore, the inpainting propagation of this LF image will be performed with the central view that was manually inpainted, which provided admissible inpainting result.

As a conclusion, the best 2D inpainting approach is to inpaint the central view either with the patch-based method or to do it manually. The diffusion-based does not present satisfactory results in most of the images, except for Figure 5.1 ("*Sill Life*" and "*Bicycle"*), and Figure A.12. It provides best results when the size of the target area is small and when there is no texture mix in the neighbour area, as seen in the bicycle region of Figure 5.1, and on the peach stem of Figure A.12. On the "*Fruits*" image, Figure A.12, the two inpainting methods can be compared in different target areas. When the target area is the yellow fruit, the patch-based provides better result. On the contrary, when the target area is the peach stem, the diffusion-based provides better result.

If the target region is present in more than one depth layer (as seen in the "*Totoro Waterfall*" image (Figure 5.2), where there is a trunk, a fountain and a bridge at different distances/layers), or if it is an object with different textures in its neighbours (for example the bowl of oranges in Figure 5.2), it is better to inpaint manually the central view. Between manually inpaint and patch-based inpaint, the manually inpaint presents (most of the times) better results (Figure 5.2, Figure A.9, and Figure A.11).

5.3. 2D-to-4D Inpainting Propagation Results

Following the 2D inpainting of the central view, this section will now focus on the 2D-to-4D propagation of those inpainting results to the whole LF image. The 2D inpainting of the central view was used by the 4D inpainting propagation algorithm to propagate the inpainting of the central view to the whole LF image (2D-to-4D propagation). These 4D inpainting propagation algorithms were implemented in LF images, and the propagation results will be analysed quantitatively and qualitatively.

5.3.1. Quantitative Results

In the quantitative results, the LF images used for testing are measured in values. Each LF image with its number of views, its resolution (vertical and horizontal number of pixels), what was the 2D inpainting algorithm used to inpaint the central view, the percentage that white pixels occupy on the mask, and the running time for each 4D inpainting propagation algorithm to propagate the inpainting of the central view to the rest of the LF image (Table 5.1).

Table 5.1 - Quantitative results of the different LF images used for tests.

LF Image	# Views	Resolution	Central view inpainting	Mask	Running Time Low Rank Matrix	Running Time Epipolar Plane	Running Time Angular Warping
Still Life	9x9	256x256	Patch-based [33]	4,4%	3,026842 s	~1 min 30 s	1,416 s
Still Life	9x9	256x256	Diffusion-based [43]	4,4%	3,095947 s	~1 min 32 s	1,419 s
Still Life	9x9	256x256	Manually [44]	4,4%	2,984581 s	~1 min 30 s	1,694 s
Bee_2	7x7	620x430	Patch-based	1,62%	3,117305 s	~2 min 02 s	3,281 s
Bee_2	7x7	620x430	Diffusion-based	1,62%	3,131780 s	~2 min 2 s	3,277 s
Bee_2	7x7	620x430	Manually	1,62%	3,070629 s	~2 min 3 s	3,269 s
Duck	7x7	620x430	Patch-based	0,50%	1,833584 s	~1 min 30 s	3,239 s
Duck	7x7	620x430	Diffusion-based	0,50%	1,830604 s	~1 min 31 s	3,242 s
Fruits V1	7x7	620x430	Patch-based	3,61%	4,849192 s	~2 min 56 s	3,546 s
Fruits V1	7x7	620x430	Diffusion-based	3,61%	4,615078 s	~3 min 14 s	3,468 s
Fruits V2	7x7	620x430	Patch-based	0,16%	1,329699 s	~1 min 22 s	3,218 s
Fruits V2	7x7	620x430	Diffusion-based	0,16%	1,291526 s	~1min 23 s	3,246 s
Toys	7x7	620x430	Patch-based	16,16%	15,555983 s	~5 min 22 s	4,534 s

Toys	7x7	620x430	Diffusion-based	16,16%	15,503111 s	~5 min 22 s	4,487 s
Toys	7x7	620x430	Manually	16,16%	15,640037 s	~5 min 26 s	4,613 s
Totoro Waterfall	11x11	620x430	Manually	15%	10,924858 s	~5 min 26 s	4,378 s
Bedroom	9x9	512x512	Patch-based	0,49%	2,442264 s	~2 min 2 s	4,644 s
Bedroom	9x9	512x512	Diffusion-based	0,49%	2,445466 s	~2 min 2 s	4,629 s
Bicycle	9x9	512x512	Patch-based	0,49%	2,367318 s	~2 min 7 s	4,677 s
Bicycle	9x9	512x512	Diffusion-based	0,49%	2,298753 s	~2 min 8 s	4,656 s
Herbs	9x9	512x512	Manually	0,90%	2,797275 s	~2 min 25 s	4,653 s

Table 5.2 - Memory used by each algorithm.

Memory Low Rank Matrix	~170 MB
Memory Epipolar Plane	~1 GB
Memory Angular Warping	~69 MB

From Table 5.1, the size of the mask influences the running time of the 4D inpainting propagation algorithms. If the percentage of white pixels (target area) on the mask is higher, the running time also increases. The algorithm that takes more time to propagate the inpainting to all viewpoints is the Epipolar Plane Diffusion (takes at least 1 minute to execute). The other two algorithms take seconds (between one to five seconds).

From Table 5.2 the algorithm that needs more memory is the Epipolar Plane Diffusion, because it has to inpainting the EPI for each viewpoint [9], and the one that uses less memory is the Angular Warping, because it is designed to be faster and more efficient inpainting of LF, and it warps directly in directions (u,v) instead on each EPI [8].

5.3.2. Qualitative Results

In the quantitative results, the LF images used for testing are described through observation. Each figure has the results from the inpainting propagation separated by the 2D inpainting method done on the central view (patch-based, diffusion-based, and manually inpainted). Then, the first row is the first inpainted viewpoint and the second row is the last inpainted viewpoint. For reference, each figure has the central view with the target area drawn in red, for the reader to understand what the area that was inpainted is.

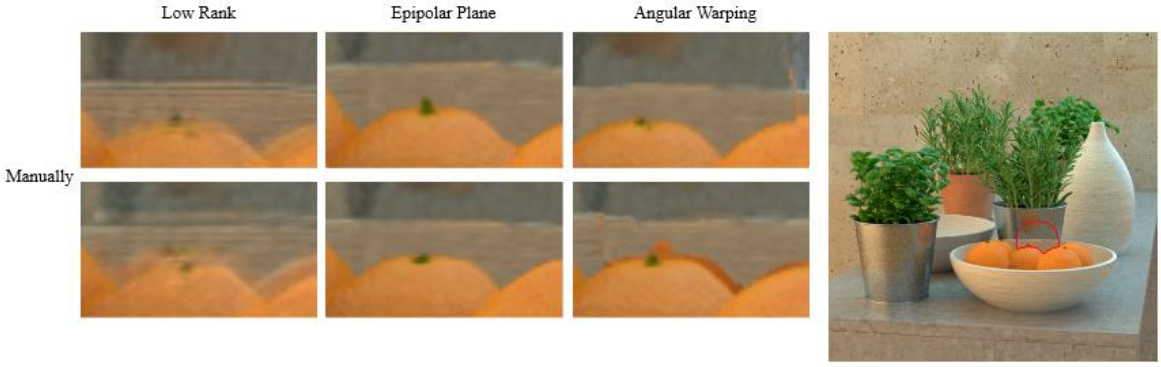


Figure 5.3 - Inpainting propagation results from the top left and bottom right views of the LF for image "*Herbs*".

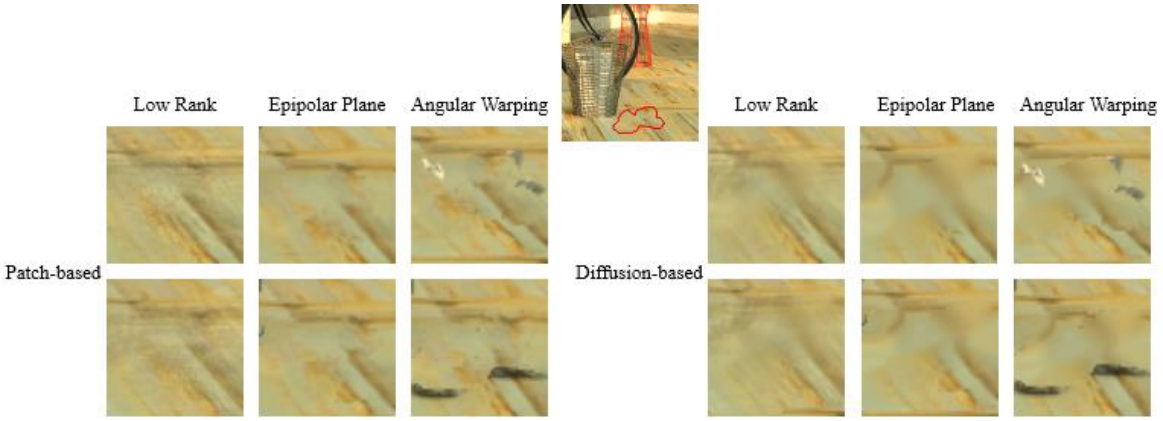


Figure 5.4 - Inpainting propagation results from the top left and bottom right views of the LF for image "*Bicycle*".

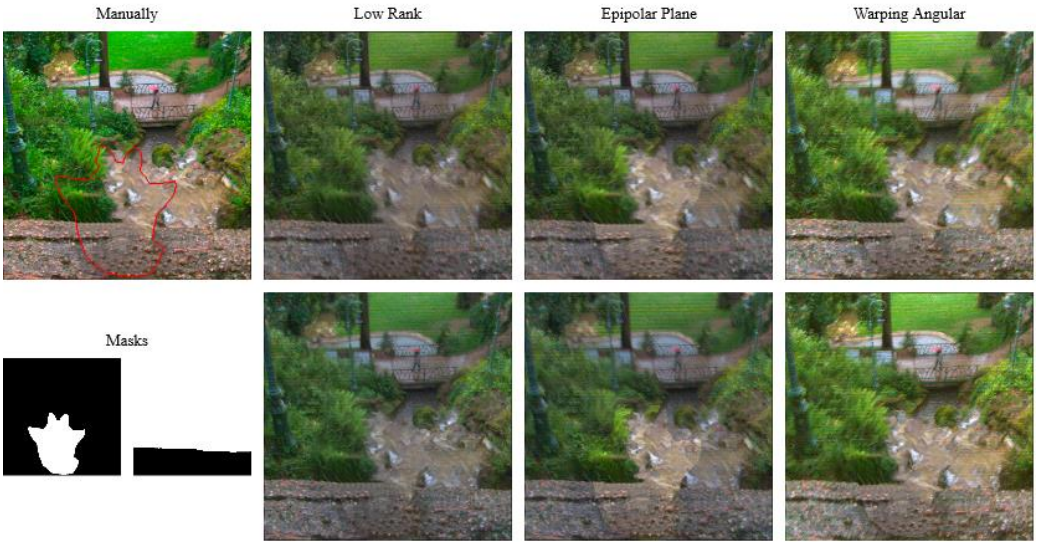


Figure 5.5 - Inpainting propagation results from the top left and bottom right views of the LF for image "*Totoro Waterfall*". This algorithm used two masks.

Figure A.14 shows the results for "*Still Life*" image inpainting propagation. The Epipolar Plane Diffusion algorithm has better propagation consistency, the pixels blend smoothly in the inpainted area because it allows coherent edit propagation, and the Low Rank algorithm presents a 'blurred' effect in the inpainted area.

In the "*Herbs*" image, the central view was inpainted manually and propagated. The Epipolar Plane Diffusion algorithm provided a more consistent and accurate propagation result than the other two (as seen in Figure 5.3), because it inpaints every EPI image, thus doing the inpainting in detail for each colour for each viewpoint.

In "*Toys*" image, the propagation provided similar results between the three algorithms (Figure A.15). According to Table 5.1, the algorithms took more time to propagate the inpainting to the whole LF image, and this image has a higher percentage of white pixels (mask area) than the other images. Nevertheless, the propagation result is similar between the three algorithms.

In Figure 5.1, the "*Bicycle*" image the best algorithm to propagate the inpainting is the Epipolar Plane Diffusion, being the worse the Angular Warping algorithm, because in this algorithm the forward warping requires disparity at the origin of the propagation, resulting in cracking effects on the image, and seen in Figure 5.4. The cracking effect happens because the local divergence of disparity does not provide information to the destination pixel.

In Figure 5.5, the image "*Totoro Waterfall*" the inpainting propagation algorithm with better results was Low Rank, also because this algorithm allows using more than one mask, which is useful if the target area is present on different depths layers, which is the case.

A test was performed for inpainting and propagation from the first view (instead of the central view), and the results can be seen in Figure A.16, Figure A.17, and Figure A.18. The results are inconsistent in Figure A.17 and Figure A.18 (on Figure A.16 one can barely notice any inconsistency). The Epipolar Plane Diffusion algorithm did not propagate the inpainting because it expects to receive the central view, and it was given the first viewpoint. It is up to each user to choose the viewpoint they want to use, knowing that they cannot use the Epipolar Plane Diffusion algorithm, and that the propagation may have inconsistencies.

In terms of conclusion, the following can be said for each 2D-to-4D inpainting propagation techniques:

- *Light field inpainting propagation via low rank matrix completion* – The Low Rank algorithm provides good propagation results when the target region is present in more than one depth layer (for example, Figure 5.5 and Figure A.13). It does not work well with the "*Still Life*" image, since the inpainted region presents a 'blurred' effect. It takes 1-3 seconds to propagate the inpainting to the whole LF data (except when the mask area is bigger than 5%), and it uses ~170 MB of memory (which is not considered a very large size).
- *Epipolar plane diffusion* – The Epipolar Plane Diffusion algorithm provides good propagation results in terms of pixel and colour blending. In the orange bowl (Figure 5.3) it is the only algorithm that generates good consistency between viewpoints, and in Figure A.14 and Figure 5.4 one can see the pixels blending more smoothly in the inpainted region (in the pear and on the tile floor). Even though it presents satisfactory inpainting propagation results, it takes more time to propagate the inpainting to the whole LF data (for example, it reaches 5 minutes for the image "*Toys*") and uses more memory than the other algorithms (~1 GB) which can be a limitation for some people/organizations/workstations.

- *Fast light field inpainting propagation using angular warping and colour-guided Disparity interpolation* – The Warping Angular algorithm provides decent propagation results in almost every LF image. The inpainting propagation is super-fast (hence the name ‘Fast light field inpainting propagation using angular warping’), even if sometimes the Low Rank algorithm is a few milliseconds faster (refer to Table 5.1). The inpainting propagation is usually consistent (except in Figure 5.3 and Figure 5.4), except for its ghosting and cracking effects. In terms of memory, it is the one that uses less memory, ~69 MB, propagating the inpainting to the whole LF image faster and consistently.

Chapter 6
Conclusions and Future Work

Light Field imaging technology is a fast-growing technology with many open topics to explore. LF inpainting is such a topic.

During the initial bibliographic search about LF inpainting, it was noticed that only very few 4D inpainting propagation algorithms exist in the literature and for most of them the code is not available for the public. For someone who wants to explore more in-depth the LF inpainting and propagation, it can be an exhausting task trying to find a code to work with. The aim for this dissertation is to explore the current LF inpainting propagation algorithms and put them into practice, developing an application for that purpose. This application contains two 2D inpainting algorithms and three 4D inpainting propagation algorithms, and the public that want to work and explore these LF inpainting and propagation algorithms can use them.

The investigation objectives of exploring the LF inpainting, formulating and implementing an easy and free-of-charge application for LF inpainting propagation were generally achieved.

The traditional existing methods for inpainting a 2D image are patch-based and diffusion-based, and they work by copying pixels from the surroundings or selecting the best patch for the target are. The 4D LF inpainting propagation algorithms can use these 2D inpainting methods. Usually, the central view is inpainted with one of these 2D inpainting methods, and then the inpainting result is propagated for the whole 4D LF image, using the 4D LF inpainting propagation algorithms.

The 4D inpainting propagation algorithms described and worked in this dissertation are 2D-to-4D inpainting propagation techniques, meaning that the inpainting of a viewpoint (normally the central view) must be the first step, and the second step is to propagate the inpainted result to the other viewpoints of the LF image.

This application contains two 2D inpainting algorithms and three 4D inpainting propagation algorithms, and they can be used by the public that want to work and explore these 2D inpainting and 4D inpainting propagation algorithms.

The developed application was developed in the computing platform MATLAB and in Linux environment. The application allows the user to import LF images or use the PlenoptiCam [14] application to create sub-aperture (viewpoints) from a dataset (images captured by Lytro cameras). After the LF image is imported, the user needs to create the mask from the target viewpoint. After providing the mask, the user needs to inpaint the target viewpoint (normally the central view) using a 2D inpainting method. Then, the final step is to choose which 4D inpainting propagation algorithm to propagate the inpainting to the whole LF image. The output (inpainted LF image) is saved, and the user can review the results, and therefore make its own conclusions and results.

For the application and the experimental results, two types of propagation techniques were used: directional propagation and low rank matrix completion. The 4D inpainting propagation algorithms used for the developed application are:

- Light field inpainting propagation via low rank matrix completion (directional propagation).
- Epipolar plane diffusion (directional propagation).
- Fast light field inpainting propagation using angular warping and colour-guided disparity interpolation (low rank matrix propagation).

In the experimental results, the patch-based method and manually inpainting the target viewpoint provided better results in 2D inpainting. Diffusion-based method can be used and works well in small target areas. There are certain images that the patch-based and diffusion-based methods did not inpaint well in the target viewpoint. In that case, the solution was to inpaint manually it with [44].

In terms of qualitative results, each 4D inpainting propagation algorithm performed the propagation differently, and each one has different features that suit different scenarios. For example, Epipolar Plane Diffusion algorithm blends smoothly the pixels in the inpainted area, but it takes a long time to propagate (at least 1 minute) and uses a lot of memory, which can be a limitation.

The Low Rank algorithm works well in target areas that are present in more than one depth layer in the original image, and memory needed to use this algorithm is approx. 170 MB.

The Angular Warping algorithm uses less memory and less running time to archive similar results from the other two algorithms. It propagates the inpainting very quickly and consistently.

To know which algorithm to use for propagation, one needs to know their features, which 2D inpainting method is the better to inpaint the target viewpoint, and if the running time and memory are a limitation. Each researcher has to investigate and choose, according to the features of each algorithm, which one is the ideal to use according to each LF image.

For future work, the application hereby developed is open-source; thence more features can be added, such as:

- More 4D inpainting propagation algorithms, such as plenopatch [7]. It implements two methods of propagation;
- More 2D inpainting algorithms for the target viewpoint, such as image melding. Image melding creates a target area to transfer various information and properties within one image to another [45];
- Other features like re-colorization and segmentation. Re-colorization can be used to correct colour fading that can happen on some viewpoints, and segmentation can be used to visualize difficult textures, complex occlusions and select automatically objects in an image;
- The application can be developed to be also used on Windows OS.

Apart from the developed application, the integrated 4D inpainting can be also developed and proposed, since there are no solutions currently of this type available in the literature, as far as the author's knowledge. This novel approach to 4D inpainting would be a good area to develop on the future: in which the 4D LF is treated as a whole and inpainting is performed on all viewpoints in an integrated way. The target area is determined in the whole 4D LF (instead of a single 2D viewpoint), and the inpainting process is done in every viewpoint simultaneously, maximizing the consistency between viewpoints.

References

[1] "Key Differences Between 4K and UHD Resolutions | RGB Spectrum."https://www.rgb.com/key-differences-between-4k-and-uhd-resolutions.

[2] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. Debevec, "A System for Acquiring, Processing, and Rendering Panoramic Light Field Stills for Virtual Reality." 2018, [Online]. Available: https://arxiv.org/abs/1810.08860.

[3] "From awkward to awesome: The top Google Pixel Magic Eraser photos." https://www.androidpolice.com/google-pixel-magic-eraser-list/.

[4] J. Bane, "Joshua Bane on Twitter: "Tryin to get the shot (pun) but your son tryin to photo bomb ? ?♂ Que the Pixel 6 Pro Magic Eraser! #PixelPostUp #teampixel #madebygoogle #giftfromgoogle @madebygoogle," May 24, 2022. https://twitter.com/JoshuaBane/status/1530670979944484864?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1530670979944484864%7Ctwgr%5E397c252756e3822180b0e9f436a38682593821a7%7Ctwcon%5Es1_&ref_url=https%3A%2F%2Fwww.androidpolice.com%2Fgoogle-pixel-m.

[5] M. Le Pendu, X. Jiang, and C. Guillemot, "Light Field Inpainting Propagation via Low Rank Matrix Completion," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 1981–1993, 2018, doi: 10.1109/TIP.2018.2791864.

[6] Williem, K. W. Shon, and I. K. Park, "Spatio-angular consistent editing framework for 4D light field images," *Multimed. Tools Appl.*, vol. 75, no. 23, pp. 16615–16631, 2016, doi: 10.1007/s11042-016-3754-y.

[7] F. L. Zhang, J. Wang, E. Shechtman, Z. Y. Zhou, J. X. Shi, and S. M. Hu, "PlenoPatch: Patch-Based Plenoptic Image Manipulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 5, pp. 1561–1573, 2017, doi: 10.1109/TVCG.2016.2532329.

[8] P. Allain, L. Guillo, and C. Guillemot, "Fast Light Field Inpainting Propagation Using Angular Warping and Color-Guided Disparity Interpolation," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11182 LNCS, pp. 559–570, 2018, doi: 10.1007/978-3-030-01449-0_47.

[9] O. Frigo and C. Guillemot, "Epipolar plane diffusion: An efficient approach for light field editing," *Br. Mach. Vis. Conf. 2017, BMVC 2017*, pp. 1–13, 2017, doi: 10.5244/c.31.47.

[10] K. W. Chen, M. H. Chang, and Y. Y. Chuang, "Light field image editing by 4D patch synthesis," *Proc. - IEEE Int. Conf. Multimed. Expo*, vol. 2015-Augus, 2015, doi: 10.1109/ICME.2015.7177447.

[11] "What is the Light Field? | LightField Forum." http://lightfield-forum.com/what-is-the-lightfield/.

[12] "Lytro Archive | LightField Forum." http://lightfield-forum.com/lytro/lytro-archive/.

[13] D. G. Dansereau, O. Pizarro, and S. B. Williams, "Decoding, calibration and rectification for lenselet-based plenoptic cameras," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 1027–1034, 2013, doi: 10.1109/CVPR.2013.137.

[14] C. Hahne and A. Aggoun, "PlenoptiCam v1.0: A light-field imaging framework," pp. 1–15, 2020, [Online]. Available: http://arxiv.org/abs/2010.11687.

[15] "MATLAB - MathWorks - MATLAB & Simulink." https://www.mathworks.com/products/matlab.html.

[16] M. Levoy, "Light fields and computational imaging," *Computer (Long. Beach. Calif).*, vol. 39, no. 8, pp. 46–55, 2006, doi: 10.1109/MC.2006.270.

[17] G. Wu *et al.*, “Light Field Image Processing: An Overview,” *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 7, pp. 926–954, 2017, doi: 10.1109/JSTSP.2017.2747126.
[18] C. Conti, L. D. Soares, and P. Nunes, “Dense Light Field Coding: A Survey,” *IEEE Access*, vol. 8, pp. 49244–49284, 2020, doi: 10.1109/ACCESS.2020.2977767.
[19] M. Levoy and P. Hanrahan, “Light Field Rendering,” 1996, doi: 10.1016/S0149-2918(01)80077-1.
[20] C. Conti, “Efficient Solutions for Light Field Coding,” 2017.
[21] B. Wilburn, “High-performance imaging with large camera arrays,” *SIGGRAPH 2006 - ACM SIGGRAPH 2006 Courses*, vol. 1, no. 212, pp. 45–113, 2006, doi: 10.1145/1185657.1185701.
[22] “The (New) Stanford Light Field Archive.” http://lightfield.stanford.edu/acq.html#gantry.
[23] G. Lippmann, “Épreuves Réversibles Donnant La Sensation Du Relief,” *J. Phys. Théorique Appliquée*, vol. 7, no. 1, pp. 821–825, 1908, doi: 10.1051/jphystap:019080070082100.
[24] “Raytrix | LightField Forum.” http://lightfield-forum.com/raytrix/.
[25] “Lytro LightField Camera | LightField Forum.” http://lightfield-forum.com/lytro/lytro-lightfield-camera/.
[26] D. E. Jacobs and M. Levoy, “Focal Stack Compositing for Depth of Field Control,” pp. 1–10, 2012.
[27] C. Guillemot and R. Farrugia, “IEEE COMSOC MMTC Communications - Frontiers Light field image processing : overview and research issues University of Malta , MALTA Fig . 1 : (left) Illustration of the two planes parameterization of the 4D static (one time instant) light field ; (ri,” vol. 12, no. 4, pp. 37–43, 2017.
[28] M. Levoy, B. Chen, V. Vaish, M. Horowitz, I. McDowall, and M. Bolas, “Synthetic aperture confocal imaging,” *ACM SIGGRAPH 2004 Pap. SIGGRAPH 2004*, pp. 825–834, 2004, doi: 10.1145/1186562.1015806.
[29] L. Wietzke, “3D Light Field Camera Risk Minimization in Endovascular Device Implantations,” 2016, [Online]. Available: www.raytrix.de.
[30] “Olivier Augereau Homepage.” http://olivier-augereau.com/inpainting.php.
[31] E. Demirağ and H. İ. Bengü, “Image Inpainting with Deep Learning,” 2021.
[32] T. T. Dang, A. Beghdadi, and M. C. Larabi, “A perceptual image completion approach based on a hierarchical optimization scheme,” *Signal Processing*, vol. 103, pp. 127–141, 2014, doi: 10.1016/j.sigpro.2013.11.036.
[33] A. Criminisi, P. Pérez, and K. Toyama, “Region filling and object removal by exemplar-based image inpainting,” *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, 2004, doi: 10.1109/TIP.2004.833105.
[34] X. Lv, X. Wang, Q. Wang, and J. Yu, “4D Light Field Segmentation from Light Field Super-Pixel Hypergraph Representation,” *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 9, pp. 3597–3610, Sep. 2021, doi: 10.1109/TVCG.2020.2982158.
[35] “Lytro Illum Dataset.” http://clim.inria.fr/IllumDatasetLF/index.html.
[36] “INRIA Light field dataset.” http://clim.inria.fr/research/LowRank2/datasets/datasets.html.
[37] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke, “A dataset and evaluation methodology for depth estimation on 4D light fields,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10113 LNCS, pp. 19–34, 2017, doi: 10.1007/978-3-319-54187-7_2.
[38] “4D Light Field Benchmark.” https://lightfield-analysis.uni-konstanz.de/.
[39] X. Jiang, M. Le Pendu, R. A. Farrugia, and C. Guillemot, “Light Field Compression with Homography-Based Low-Rank Approximation,” *IEEE J. Sel. Top. Signal Process.*, vol.

11, no. 7, pp. 1132–1145, 2017, doi: 10.1109/JSTSP.2017.2747078.
[40] “Create customizable freehand ROI with assistance from object edges - MATLAB drawassisted.” https://www.mathworks.com/help/images/ref/drawassisted.html.
[41] “Restore specific image regions using exemplar-based image inpainting - MATLAB inpaintExemplar.” https://www.mathworks.com/help/images/ref/inpaintexemplar.html.
[42] “Restore specific image regions using coherence transport based image inpainting - MATLAB inpaintCoherent.” https://www.mathworks.com/help/images/ref/inpaintcoherent.html.
[43] F. Bornemann and T. März, “Fast image inpainting based on coherence transport,” *J. Math. Imaging Vis.*, vol. 28, no. 3, pp. 259–278, 2007, doi: 10.1007/s10851-007-0017-6.
[44] M. Daisy *et al.*, “A Fast Spatial Patch Blending Algorithm for Artefact Reduction in Pattern-based Image Inpainting To cite this version : HAL Id : hal-00932281 A Fast Spatial Patch Blending Algorithm for Artefact Reduction in Pattern-based Image Inpainting,” 2014.
[45] S. Darabi, E. Shechtman, C. Barnes, Dan B Goldman, and P. Sen, “Image melding: Combining inconsistent images using patch-based synthesis,” 2012. doi: 10.1145/2185520.2185578.

Appendix A
Light Field Test Content

The LF test content that was used throughout this dissertation is illustrated in this appendix.

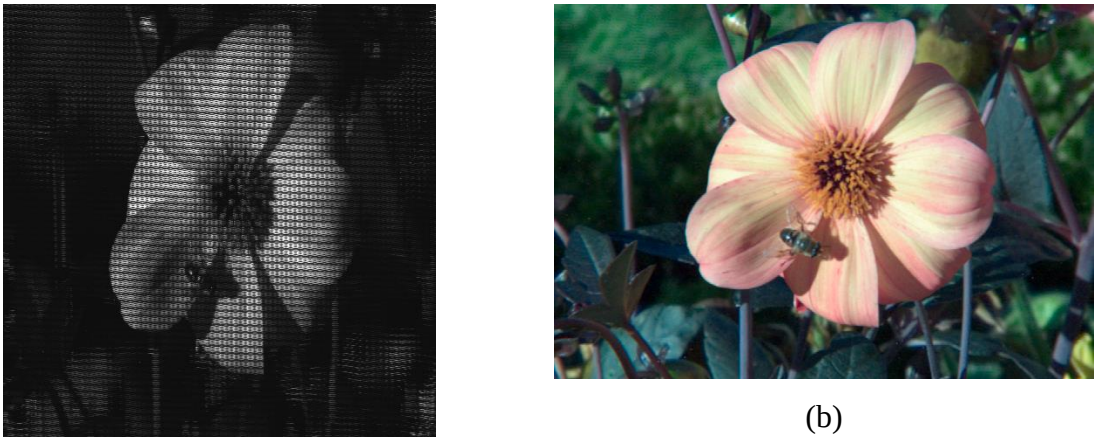
The content is here separated into LF images, presented in Section 4.1; inpainting results, presented in Section 5.2; and inpainting propagation results, presented in Section 5.3.

A.1. LF Test Images

Figure A.1 - Extracting viewpoints with the toolbox [13] produces challenging data with noise and colour and illumination variations between viewpoints. Row (a) represents the first three viewpoints, the row (b) represents the last three viewpoints. Comparing the three datasets used for this application. The column (a) images are the first viewpoints of the LF image, the column (b) images are the central view of the LF image, and the column (c) images are the last viewpoints of the "*Totoro Waterfall*" LF image.

(a) (b)

Figure A.2 - Two viewpoints of the "*Totoro Waterfall*" image (out of 121 viewpoints)).



(a) (b)

Figure A.3 - Raw image file (a) and central view (b) of "*Bee_2*" LF image.

(a)

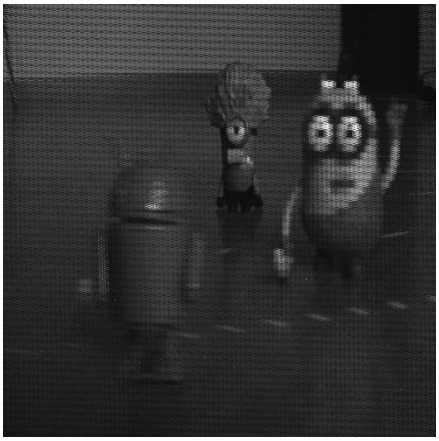
(b)

Figure A.4 - Raw image file (a) and central view (b) of "*Duck*" LF image.

(a)

(b)

Figure A.5 - Raw image file (a) and central view (b) of "*Fruits*" LF image.



(a)

(b)

Figure A.6 - Raw image file (a) and central view (b) of "*Toys*" LF image.

(a) (b) (c)

Figure A.7 - Central view (a) and two viewpoints (b) and (c) of "*Bicycle*" LF image.

(a) (b) (c)

Figure A.8 - Central view (a) and two viewpoints (b) and (c) of "*Herbs*" LF image.

A.2. Inpainting Results

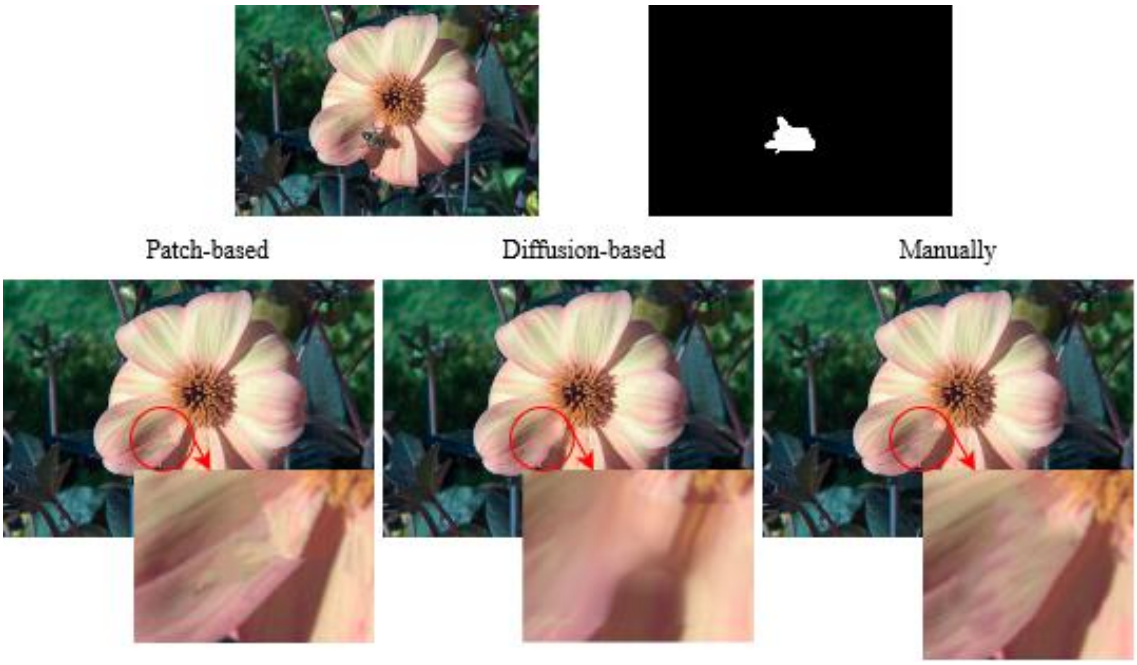


Figure A.9 - Inpainting of the central view of the image "*Bee_2*" with three different inpainting methods.

Figure A.10 - Inpainting of the central view of the image "*Duck*" with two different inpainting methods.

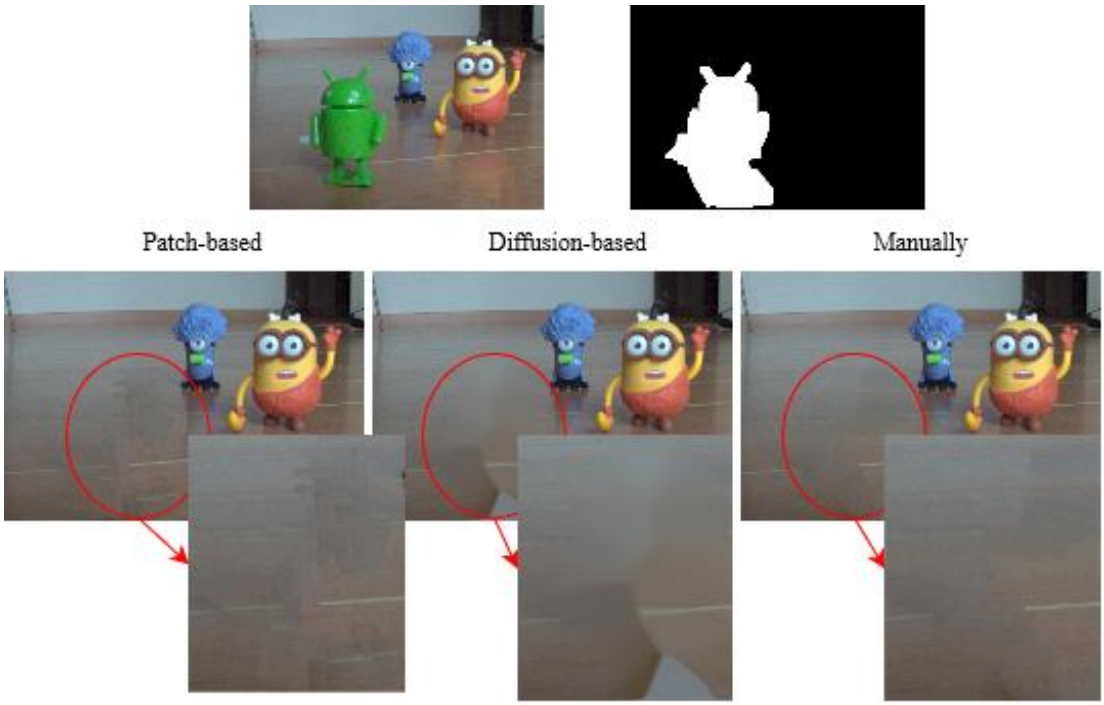


Figure A.11 - Inpainting of the central view of the image "*Toys*" with three different inpainting methods.

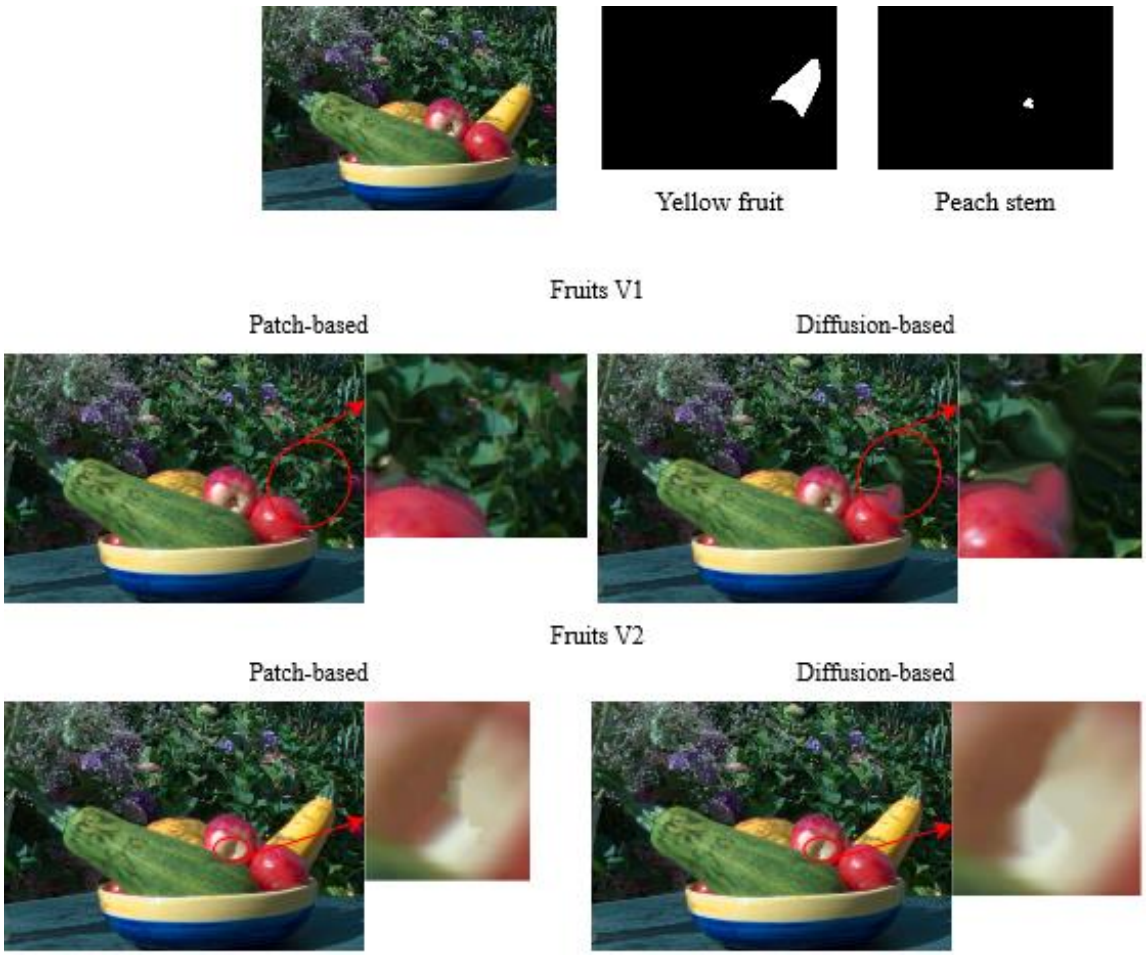


Figure A.12 - Inpainting of the central view of the image "*Fruits*" with two different inpainting methods. The "*Fruits V1*" inpaints the region of the yellow fruit. The "*Fruits V2*" inpaints the region of the peach stem.

A.3. Inpainting Propagation Results

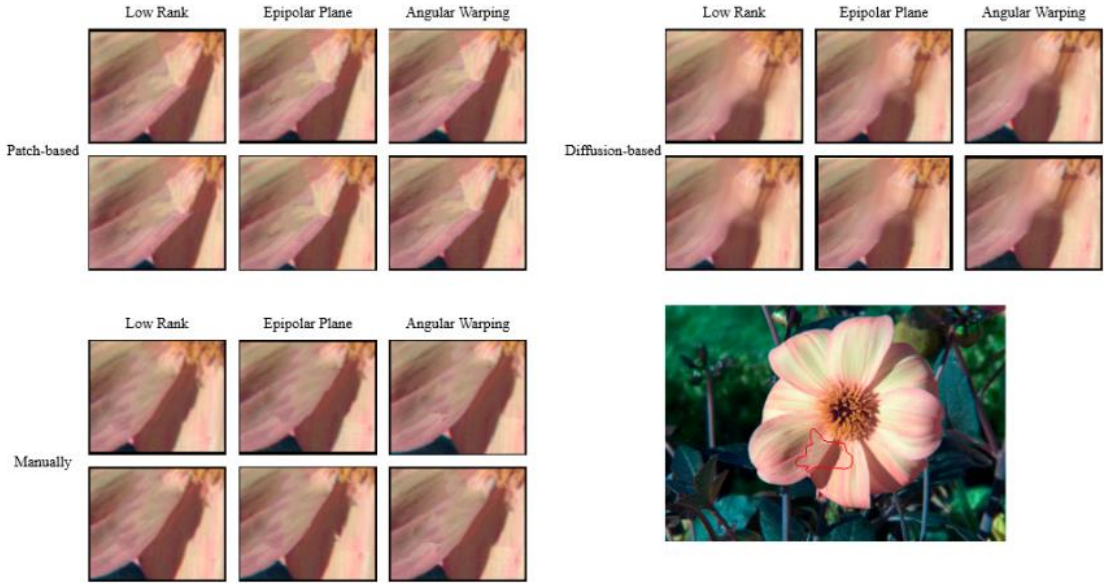


Figure A.13 - Inpainting propagation results from the first viewpoint and last viewpoint of the LF for image "*Bee_2*". The mask contour of the target region is also displayed.

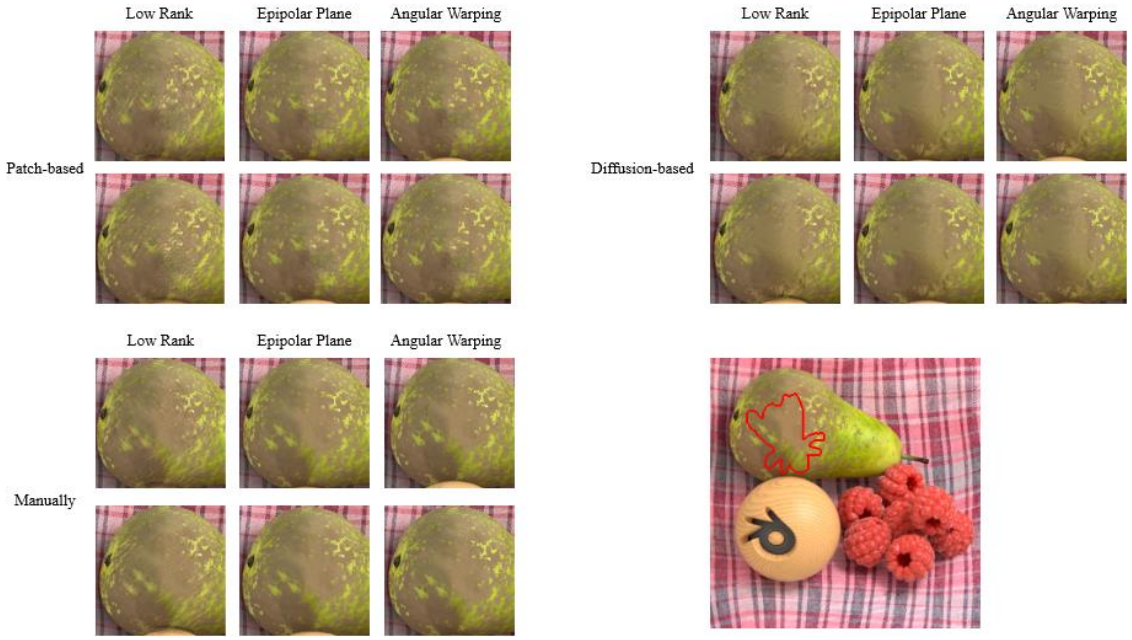


Figure A.14 - Inpainting propagation results from the first viewpoint and last viewpoint of the LF for image "*Still Life*". The mask contour of the target region is also displayed.

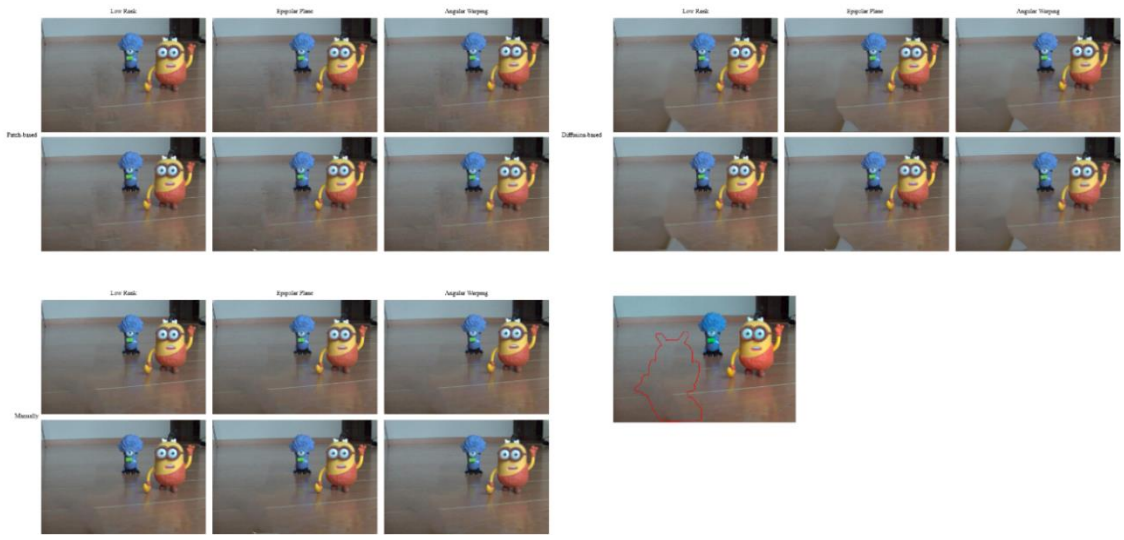


Figure A.15 - Inpainting propagation results from the first viewpoint and last viewpoint of the LF for image "*Toys*". The mask contour of the target region is also displayed.

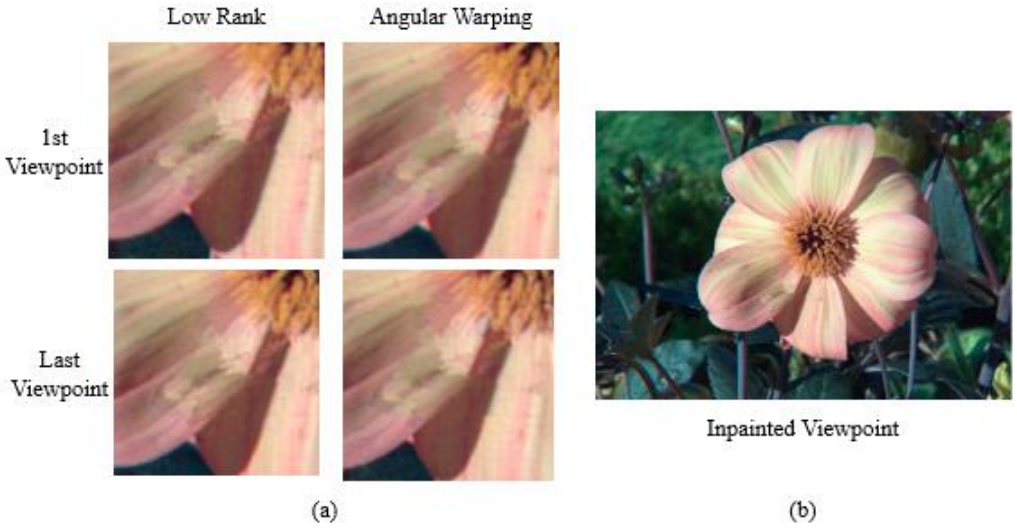


Figure A.16 - Inpainting propagation results of the LF for image "*Bee_2*" (a). The inpainting was done with the patch-based inpainting algorithm (b), and the selected viewpoint was first viewpoint, instead of the central view.

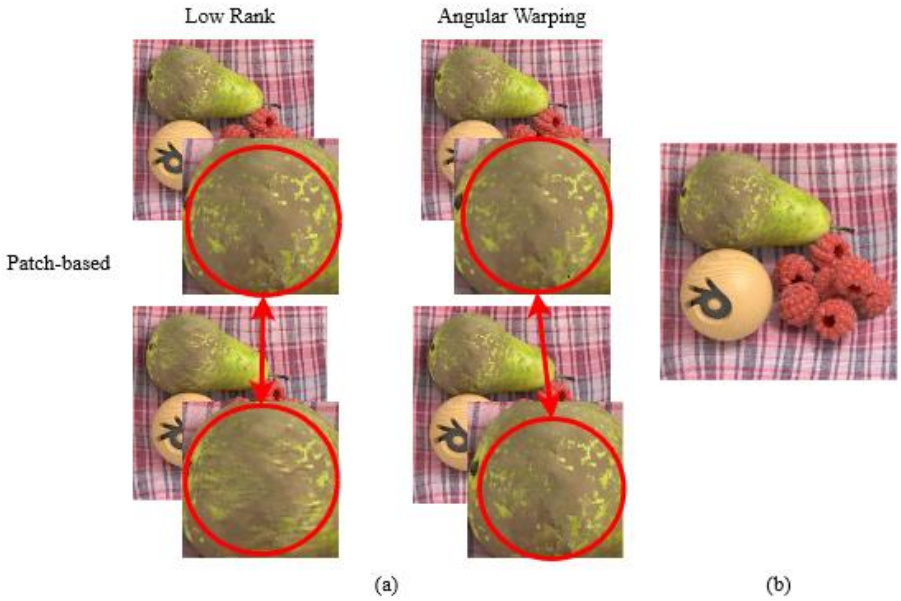


Figure A.17 - Inpainting propagation results of the LF for image "*Still Life*" (a). The inpainting was done with the patch-based inpainting algorithm (b), and the selected viewpoint was first viewpoint, instead of the central view.

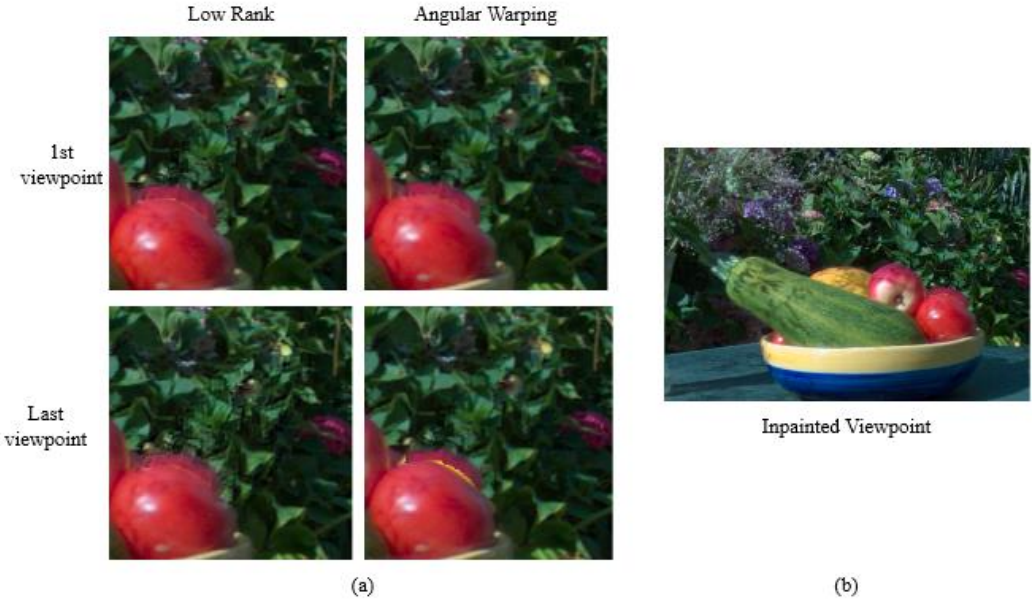


Figure A.18 - Inpainting propagation results of the LF for image "*Fruits*" (a). The inpainting was done with the patch-based inpainting algorithm (b), and the selected viewpoint was first viewpoint, instead of the central view.