
Technische Universität Berlin

Master's Thesis

# Fast Sparse Light Field Reconstruction with Shearlet-based Inpainting

*Author:*
Héctor Andrade Loarca

*Supervisor:* Prof. Dr. Gitta Kutyniok
*Second reader:* Prof. Dr. Reinhold Schneider.

Fakultät II
Institut für Mathematik
AG Angewandte Funktionalanalysis
11. September 2017

## Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbständige und eigenständige Anfertigung versichert an Eides statt: Berlin, den

4. August 2017

Héctor Andrade Loarca

# Zusammenfassung in deutscher Sprache

## Schnelle Rekonstruktion für Dünne Lichtfelder mit Shearlet-basierten Einfärbungen

Diese These ist angewendete

*A Natasha y los años que nos quedan juntos*
*A mi madre Julieta y Padre Héctor*
*sin los cuales nada de esto hubiera pasado*
*A Patricia, Sara y Cristina,*
*por enseñarme cada día lo que es una familia*

An article about computational result is
advertising, not scholarship. The actual
scholarship is the full software environment,
code and data, that
produce the result

*Buckheit and Donoho (1995)*

# Acknowledgements

To my mom.
To all of you, thank you very much.

Berlin, September 2017

# Contents

**1 Introduction** **1**

**2 Light Field Photography** **3**
2.1 Light Field Photography in the History . . . . . 5
2.2 Light Field Acquisition Settings . . . . . 6
2.3 Typical applications for the Light Field Theory . . . . . 9
2.4 Geometric proxy: Stereo Vision and multiview Epipolar Geometry . . . 11
2.4.1 Epipolar constraint . . . . . 11
2.4.2 Bolles feature tracking technique and experimental setup . . . . 13
2.4.3 Functional analysis approach to EPI . . . . . 15
2.4.4 Geometrical Approach to EPI . . . . . 15
2.5 Physical and computational setup for sparse acquisition of epipolar plane 17
2.5.1 Physical setup and sampling rate . . . . . 17
2.5.2 Followed pipeline . . . . . 18
2.5.3 Geometric construction of epipolar lines . . . . . 20
2.5.4 Tracking point algorithms . . . . . 22
2.5.5 Procedure for tracking and painting the EPIs . . . . . 27

**3 Shearlets** **33**
3.1 Shearlet Systems and Transform . . . . . 39
3.2 Shearlets as Frames . . . . . 46
3.3 Universal Shearlets and Alpha Shearlets . . . . . 48
3.4 Image inpainting using Shearlet Parseval Frames . . . . . 56
3.5 0-Shearlets . . . . . 61

**4 Inpainting Sparse Sampled Epipolar-plane and Computing Depth Map 67**
4.1 Iterative thresholding algorithm for EPIs inpainting using 0-Shearlets . 67
4.2 Results of sparse EPIs inpaiting . . . . . 70
4.3 Line detection in inpainted EPIs and depth map computation . . . . . 76
4.4 Performance and results comparison with previews works . . . . . 88

**5 Conclusion and outlook** **97**

**Appendices** **99**

**A Code for point tracking** **101**

**B Code for painting EPIs** **105**

**C Code for inpainting sparse EPIs** **109**

**D Code for computing the depth map** **113**

# Chapter 1

# Introduction

Introduction template.

# Chapter 2

# Light Field Photography

The propagation of the light rays in the 3D space can be completely described by a 7D continuous function $R(\theta, \phi, \lambda, \tau, V_x, V_y, V_z)$, where $(V_x, V_y, V_z)$ is a location in the 3D space, $(\theta, \phi)$ are propagation angles, $\lambda$ is the wavelength of the corresponding light wave and $\tau$ the time; this function is known as the plenoptic function and describes the amount of light flowing in every direction through every point in space at any time, the magnitude of $R$ is known as the radiance. In an 1846 lecture entitled "Thoughts on Ray Vibrations" Michael Faraday proposed for the first time that light could be interpreted as a field, inspired by his work on magnetic fields; but the idea of a plenoptic function representing the spectral radiance distribution of rays was first proposed by Adelson and Bergen [4].

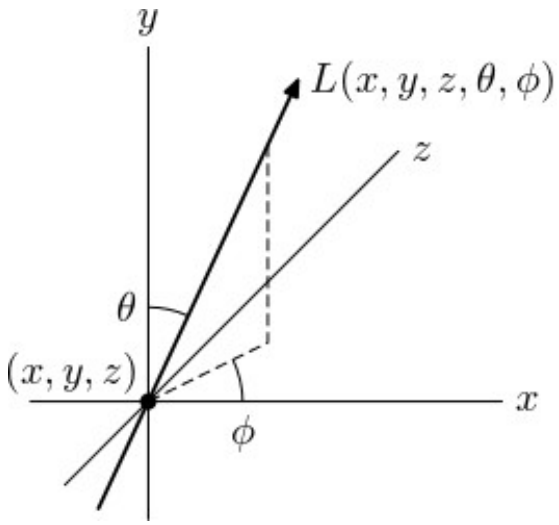



FIGURE 2.1: Spatio-angular parametrization of the plenoptic function for fixed $\tau$ and $\lambda$. Figure taken from Wikipedia (https://en.wikipedia.org/wiki/Light_field)

In a more practical approach the plenoptic function can be simplified to a 4D version, called 4D Light Field or simply Light Field (abbreviated from now on as LF), denoted as the function $L_4$. The LF quantifies the intensity of static and monochromatic light rays propagating in half space, though this seems like an important reduction of information, this constraint does not substantially limit us in the accurate 3D description of the scene from where the light rays come from. The Light Field of a ·D scene (subspace) describes the 3D information like the depth of the points in the scene and can be recovered by taking multiple pictures (views) of the scene. The goal of this thesis is to explain a particular fast technique for this purpose.

There exists three typical forms of this 4D approximation:

1. The LF rays positions are indexed by their Cartesian coordinates on two parallel planes, also called the two-plane parametrization $L_4(u, v, s, t)$.

2. The LF rays positions are indexed by their Cartesian coordinates on a plane and the directional angles leaving each point, $L_4(u, v, \phi, \theta)$.

3. Pairs of points on the surface of a sphere $L_4(\phi_1, \theta_1, \phi_2, \theta_2)$.

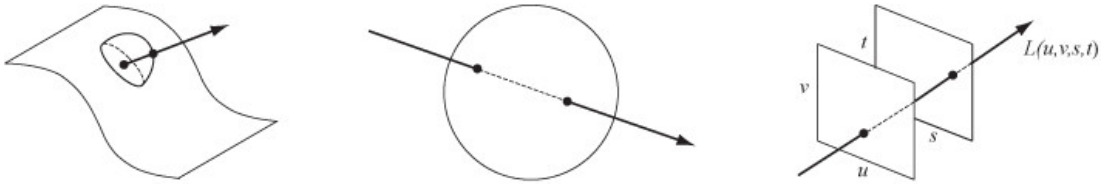



FIGURE 2.2: Three different representations of 4D LF. Left: $L_4(u, v, \phi, \theta)$. Center: $L_4(\phi_1, \theta_1, \phi_2, \theta_2)$. Right: $L_4(u, v, s, t)$. Figure taken from Wikipedia (https://en.wikipedia.org/wiki/Light_field)

In this work we will centered our attention in the two plane parametrization $L_4(u, v, s, t)$, if you are interested in the other descriptions we recommend to see [5]. In order to understand deeply this way of LF description, lets consider a camera with image plane coordinates $(u, v)$ and the focal distance $f$ moving along the $(s, t)$ plane.

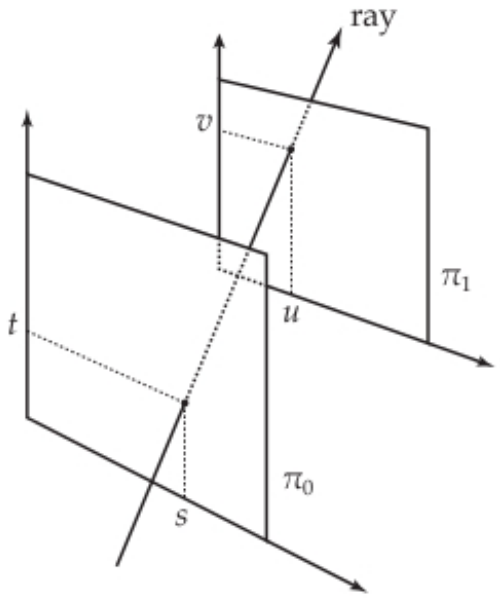



FIGURE 2.3: Graphic representation of the two plane parametrization of a single ray on the LF which is parametrized by the intersection $(s, t)$ and $(u, v)$ with planes $\pi_0$ and $\pi_1$, respectively. Figure taken from [6] p.21

For simplicity one can constrain the vertical camera motion by fixing $s = s_0$ and moving the camera along the $t - axis$ in an straight light motion, in the section 2.4 we

will see that this constraint leads to an elegant geometric 3D representation of the scene called Epipolar Geometry, this multiview aquisition is refered as parallax only (HPO). Under this constraint, images captured by successive camera positions $t_1$, $t_2$,... can be stacked together, and one can also interpret each camera position as a time step.

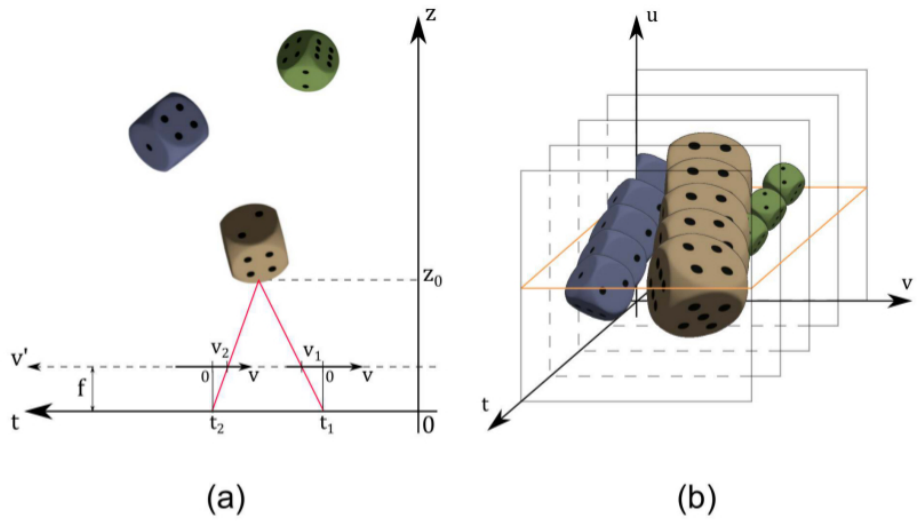



FIGURE 2.4: Stacked captured images represented in (b) from the scene setup (a). Figure taken from [3] p. 2

## 2.1 Light Field Photography in the History

For different reasons of interest for science and art, capturing light fields has been an active research area for more than 110 years (the reason will be explained in detail on the section 2.3). In 1903 Herbert E. Ives [7] was the first to realize that the light field inside a camera can be recorded by placing a pinhole or lenslet array in front of a film sensor (what is know as pinhole camera). On the other hand, in 1908 the french physicist and Nobel laureate Gabriel Lippmann published two articles about something that he called *photographie intégrale* (translated as integral photography) [8] in which he describes an imaging apparatus with an arrange of small lenses on a 2D grid that are able to capture multiple images of a scene with viewpoint variations; the captured scene is reproduced in 3D as the viewer sees the parallax while the viewpoint changes. Is quite surprising that almost 110 year ago he could have the idea that modern state of the art LF aquisition systems use nowadays.

Even some experiments to aquire the Light Field of a static scene were already proposed since the beginning of the XX century, the first contribution on the mathematical formalization of the Light Field Theory were proposed in 1991, when Adelson and Bergen [9] found a way to systematically categorize the visual elements in *early vision*, which in combination, form the visual information of the world. Here by *early vision* we mean the processes that are involved in the first steps of the visual cortex, namely, basic segmentation, shape detection and motion analysis between others (for further information about *early vision* [10] is highly recommended); so with this purpose Adelson and Bergen defined the *plenoptic function* which was already discussed at the beginning of this Chapter.

The history of Light Field Theory can be separated in the three main steps in the study of the Light Field: The acquisition, the processing (which include in the most of the cases a geometry proxy) and the rendering, which are closely related and vary in computational complexity and for which there exist plenty of different approaches; in this thesis we will center our study on the first two steps.

It is also worth to mention that in the last decade two companies had manufactured Cameras that are able to capture the 4D Light Field, also known as plenoptic cameras; the first was Raytrix founded by the german computer scientists Christian Perwass and Lennart Wietzke that released their camera in 2010 mostly focused on industrial application on 3D reconstruction (one can see their paper [12] for a good reference) rather than general consumers. Later in 2012 the american company Lytro came out with a plenoptic camera that was the first consumer light field camera for the general public, that has as a principal feature the possibility to do refocusing in the pictures taken by the camera (as a reference for this camera we recommend to read the Stanford Technical Report written by the CEO of the company Ren Ng [13]). Both companies produce cameras that capture the light field using an array of lenses, this and other LF acquisition setting will be discuss in the next section.


FIGURE 2.5: Industrial plenoptic camera Raytrix R11, produced by Raytrix. Figure taken from `https://petapixel.com/assets/uploads/2010/09/raytrix.jpg`

## 2.2 Light Field Acquisition Settings

The first creativity step in the experimental study of Light Field is the form of acquisition; using only our physical intuition is not trivial to come up with an idea of a system that captures faithfully the Light Field coming from a static scene that will be able to be processed by some straight-forward algorithm. From the beginning of the last century until today, scientists, engineers and hobbyists have proposed different approaches for the Light Field Acquisition Settings.

As we have seen in the last section, the first attempts of settings were the pinhole camera proposed by Ives and the lenslet array proposed by Lippmann. The next variation of setting was proposed more than eighty years later by Adelson and Wang [14] that in 1992 using the theory of plenoptic function (developed by Adelson itself) presented a design of a plenoptic camera where the light rays that pass through the main


FIGURE 2.6: Consumer plenoptica camera Lytro Illum, produced by Lytro. Figure taken from `https://www.ephotozine.com/articles/lytro-illum-review-26434/images/highres-Lytro-Illum-6_1414410926.jpg`

lens are recorded sparately using a lenticular array placed on the sensor plane, they used the light field recorded with this camera to obtain the scene depth by analyzing the directional variation of the radiance captured in the image; this is basically the Lippmann design but applied to digital cameras. Ng et al. [13] from Lytro used the same design of Adelson and Wang to produce the Lytro cameras.

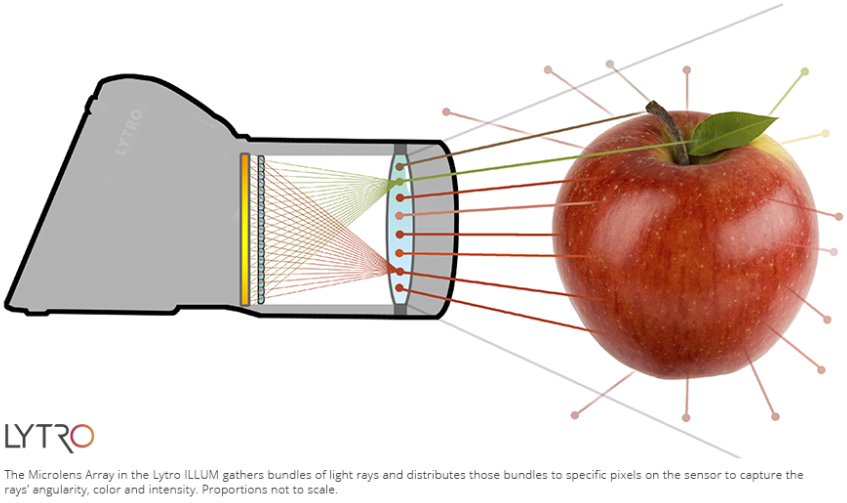



FIGURE 2.7: Diagram of Adelson and Wang design in Lytro cameras. Figure taken from `https://s3.amazonaws.com/lytro-corp-assets/blog/Lytro_ILLUM.png`

In 2006 Joshi et all [15] used a one-dimensional camera array and a motorized stage for for their real-time matting system. This technique of multicameras/multiviews acquisition is also quite common with camera arrays varying in position and size. The approach followed in this thesis take this technique as acquisition setting, the actual setup used will be explained in more detail in the section 2.5. Even this design can be

built without having an a priori designed optics which is an advantage over the Lytro camera, it has also a disadvantage since this setup can be very bulky (mechanical tracks are generally big and heavy) and therefore not hand held as the Lytro.

A less bulky approach are the ones with Light-modulating codes in mask-based systems, that use coded masks in front of lenses for coded acquisition of the scene. Veeraraghavan et al. [16] where the first implementing a coded aperture technique to computatinally demultiplex the light rays collected through the camera's main lens. This attempt is less bulky than the multicameras and more light efficient than the pinhole arrays but it sacrifices image resolution, since the number of sensor pixels is the upper limit of the number of light rays captured (problem than camera arrays and lenslets does not have). To overcome this problem, Wetzstein et al. [17], analized multiplexing light fields onto a 2D image sensor and developed a theory for multiplexing and a computational reconstruction algorithm.

A significant challenge of acquisition is that the captured set of images is very data-intensive and also redundant, mostly when one tries to recover high resolution light field form images with resolution above $(2000px)^2$. In order to tackle this issue, since the early papers on Light Field, the discussion about compact or sparse representation and compression schemes have played an important role in the area. For instance, Levoy and Hanrahan [18] proposed in 1995 several representations for 4D light fields and apply a lossy vector quantization followed by entropy coding; whereas Gortler et al. [19] in the same year applied standard image compression like JPEG to some of the views and pointed out the importance of depth information for sparser representation.

Later on, Wetzstein with the Camera Culture Group of the MIT Media Lab developed a compressive light field camera architecture that allows the light fields to be recovered from a single picture with higher resolution than previously possible, using three main components: light field atoms as a sparse representation of natural light fields (that involves dictionary learning with light field atoms as elements), and optical design that allows for capturing optimized 2D light field projection also based in the coded masks technique, and robust sparse reconstruction methods to recover a 4D ligth field from a single coded 2D projection. In our opinion, even this approach allows us to get a very high resolution of light fields, is a trade off by its requirements of high performance computation and its limitations coming from the baised learned dictionaries from a limited set of scenes.

In the multicameras instance, Vargharshakyan et al. [3] developed in 2015 an image based rendering technique based on light field reconstruction from a limited set of perspective views acquired by cameras, which in that sense is compressed. Even the preprint was presented in 2015, the actual paper was just published this year and it represents a state of the art light field recovery technique. The technique utilizes sparse representation of epipolar-plane images (a very important concept in stereo-vision that will be explained carefully in section 2.4) using as sparsifying system, adapted shearlet transform. This compressive approach was used in this thesis for the light field recovery and we picked it since we consider it as very interesting mathematically since it uses geometry (epipolar-plane representation), compressed sensing (sparse recovery), computer vision (feature point tracking and straight line detection) and functional analysis (shearlet representation). In the next sections and chapters we will cover every detail regarding the technique hoping it is comprehensive for everybody with no expert knowledge of any of the areas but just basic concepts.


FIGURE 2.8: Single coded 2D projection from the work of Wetzstein, Figure taken from [11] p. 8

## 2.3 Typical applications for the Light Field Theory

We already introduced the concept of 4D Light Field, how this concept has been developed through more than a century already and some techniques of acquisition, but one fundamental question arises; what is the interest of studying Light Fields?, and this question has many answers. Of course the first one is just interest on the mathematical foundation of Early Vision, but this allow us not just to understand more the way the human brain works for vision interpretation but also to enhance the quality of information of certain spatial scene. For a more clear exposition we will enumarate some of the more remarkable applications of light field recovery:

- **Illumination engineering**: With the study of the ligth field one can derive in a closed form the illumination patterns that would be observed on surfaces due to ligth sources of various shapes positioned above these surface.

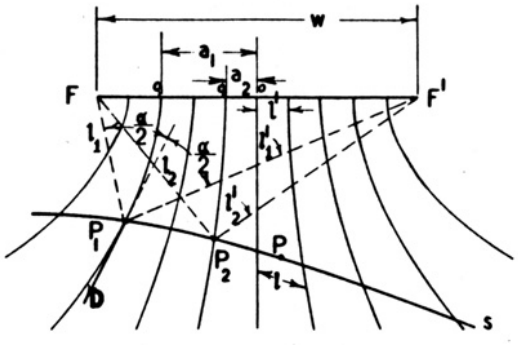



FIGURE 2.9: Downward-facing light source which induces a light field whose irradiance vectors curve otwards, Figure taken from `https://en.wikipedia.org/wiki/File:Gershun-light-field-fig24.png`

- **View synthesis or rendering**: One of the most visible applications of the light fields, which centers of the synthesis of intermediate views from a given set of captured views of a 3D visual scene, also called image-based rendering. Immersive

visual applications as free viewpoint television and virtual reality require a dense set of images of a scene, but the scene is typically captured by a limited number of cameras that form a coarse set of multiview images. Modern methods for view synthesis are based in two different approaches: estimation of the scene depth and synthesis of novel views based on the estimated depth and the given images, where the depth works as correspondence map for view reprojection (something that could be interpreted as inverse projection or triangulation). The limitation on this approach is that the quality of depth estimation is dependent on the scene content, causing visually annoying artifacts in the rendered (synthesized) views when the depth map has small deviations (for further information of this one can read [20]).

The best approach so far that fixes this problem is based on the concept of plenoptic function and its light field approximation. The scene capture and intermediate view synthesis problem can be formulated as sampling and consecutive reconstruction (interpolation) of the underlying plenoptic function. LF based methods consider each pixel of the given views as a sample of a multidimensional LF function, thus the unknown views are function values that can be determined after its reconstruction from samples.

- **Synthetic aperture photography (Light Field rendering)**: One can approximate the view that would captured by a camera having a finite aperture (non-pinhole) when integrating an appropiate 4D subset of the samples in a light field. This view has a finite depth of field (one can focuse until a finite depth on the scene). One can focus on different fronto-parallel or oblique planes in the scen by shearing the light field before performing this integration (one can check [21] for the fronto-parallel case). Like in the case of Lytro cameras, this permits its photographies to be refocused after they are taken.

- **3D display**: One can present a light field using technology that maps each sample to the appropiate ray in physical space, one obtains then an autostereoscopic visual effect when viewing the original scene (hologram-wise). With non digital technologies to do this, one can use holography; digital technologies of 3D display include placing an array of lenslets over a high-resolution display screen, or projecting the imagery onto an array of lenslets using an array of video projectors; if this last one is combined with an array of video cameras, one can capture and display a time-varying light field, which basically constitutes a 3D television system (check [22]).

- **Light Field microscopy**: Light field permit manipulation of viewpoint and focus after the imagery has been recorded. By inserting a microlens array into the optical train of a conventional microscope, one can capture light fields of biological specimens in a single photograph. The ability to create focal stacks from a single photograph allows moving or light-sensitive specimens to be recorded, with 3D deconvolution one can produce a set of cross sections which can be visualized using volume rendering, one very recommended reference on that sense is [23].

- **Brain imaging**: Neural activity can be recorded optically by genetically encoding neurons with reversible fluorescent markers that indicate the presence of calcium ions in real time. Since Light field microscopy captures full volume information in a singel frame, it is possible to monitor neural activity in many individual neurons randomly distributed in a large volume at video framerate.

A quantitative measurement of neural activity can even be done despite optical aberrations in brain tissue and without reconstructing a volume image [24].

- **Glare reduction**: Glare is a difficulty seeing in the presence of bright light such as direct or reflected light, and arises due to multiple scattering of light inside the camera's body and lens optics and reduces image contrast. While glare has been analyzed in 2D image space, it is useful to identify it as a 4D ray-space phenomenon [25]. By analyzing the ray-space inside a plenoptic camera, one can classify and remove glare artifacts, since in ray-space glare behaves as high frequency noise and can be reduced by outlier rejection (for instance thresholding). This application represents a great solution for some issues in film postproduction.

We think this examples of application make very clear the important role that Light Field recovery plays in technology, medicine and art; therefore we also think that is worth to study new optimal methods for this recovery.

## 2.4 Geometric proxy: Stereo Vision and multiview Epipolar Geometry

3D geometry reconstruction has been an interest of study for decades and there is a plenty of material where one can look at, where many different approaches are presented. One of the first approaches to recover depth information from a dense sequence of images is the seminal work of Bolles et al. [26] a very recommended classic in the topic; though its rendering technique is old and not robust enough for a dense reconstruction of scenes with occlusions, vary illumination and other features; one can use the geometric approach to obtain underlying linear structures of the light field. Due the mathematical simplicity and straight forward implementation of this approach we used this model to approach the Epipolar-plane images of the 3D scene to reconstruct the 4D Light Field, but in this case we have a sparse sequence of images so we used a sparse representation for the epipolar plane to tackle this issue.

### 2.4.1 Epipolar constraint

One of the fundamental tasks of computer vision is to describe a scene in terms of coherent threedimensional objects and their spatial relationships. This tasks present clear limitations for two main reasons:

- There is an enormous diversity of objects and an almost limitless ways in which they can occur in scenes.
- Classical images have an inherent ambiguity; since the process of forming an image captures only two of the three dimensions of the scene, an infinity of three-dimensional scenes can give rise to the same two-dimensional image; therefore no single two-dimensional image contains enough information to enable reconstruction of the three-dimensional scen that gave rise to it.

Human vision tackles this limitation with the use of knowledge of the scene objects and multiple images, like stereo pairs and image sequences acquired by a moving observer; though the mathematical and computational implementation of this features is not trivial but using more than one image makes it theoritically possible, under certain circumstances (that go from position of the views to sampling rate) modern techniques on stereo vision have made this possible up to some precision. As we already mention in this thesis we will make use of the epipolar plane image analysis technique.

The epipolar plane image analysis proposed by Bolles [26] is a technique to make a threedimensional description of a static scene from a dense sequence of images; the sequence is dense in the sense that its images form a solid block of data in which the temporal continouity from image to image is equal to the spatial continuity (namely the resolution of the picture). Slices of this block encode the 3D position of objects and occlusion of an object by another.

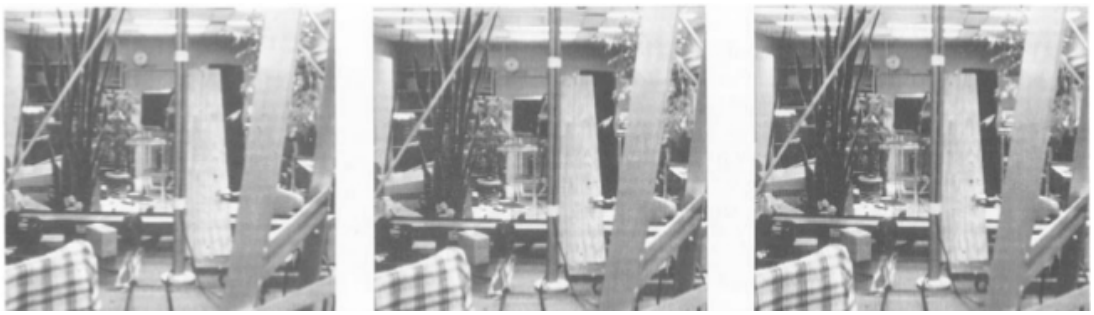


FIGURE 2.10: First three of 125 images taken by Bolles et al. Figure taken from [26] p. 16

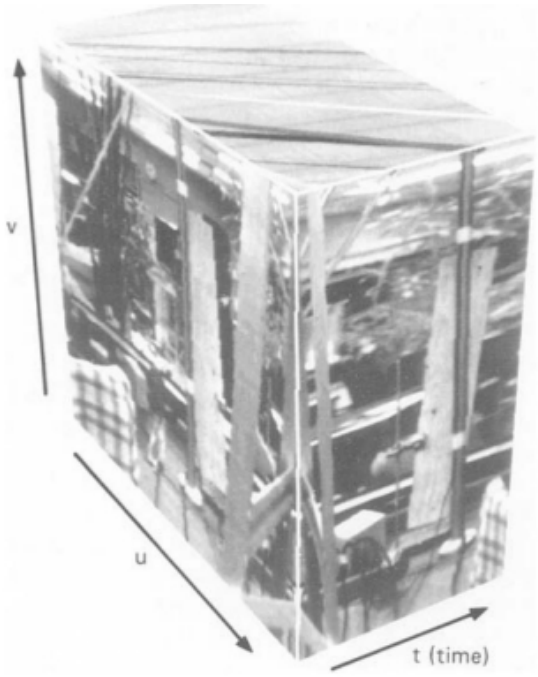



FIGURE 2.11: Spatiotemporal solid of data corresponding to the sequence on the Figure 2.10, Figure taken from [26] p.16

One can supply the separate analysis of both camera motion and object position by an unified treatment of parameters and concentrate solely in object positions while knowing the camera followed trajectory, this known motion assumption is appropriate for autonomous vehicles with intertial-guidance systems and some industrial tasks. This assumption is called the **"epipolar"** constraint and its most important feature is that it reduces the search required to find matching features from two dimensions to one and is derived from the known position of one camera view with respect to the other.

The epipolar constraint as we just mentioned reduces the complexity of matching features between successive images (by search dimensional reduction), even though matching features still one of the most difficult steps in motion processing. In stereo analysis, it is well known that the difficulty of finding matches increases with the distance between the lens centers of successive views, so as a second assumption we suppose that the images were taken very close together.

At the time that Bolles et al. developed the epipolar plane image analysis technique matching features was indeed a very complex task to implement; they did not have digital cameras of high resolution as today, and also the most common algorithms on feature extraction/tracking for motion flow were developed after 1988 (we will discuss in detail about this on the section 2.5) just one year after Bolles proposed this approach. For this reason they developed their own very creative way to track features that is worth to mention shortly, to be able to compare with the modern robust algorithms.

### 2.4.2 Bolles feature tracking technique and experimental setup

Bolles and his group in the Artificial Intelligence Center at Menlo Park developed as a feature tracker an edge detection and classification technique for analyzing one slice of the data (spatio-dimensional block of images) at a time. For this end, they adapted this approach to a range sensor, which gathered hundreds of slices in sequence. The sensor, a standard structured-light sensor, projected a plane of light onto the objects in the scene and then triangulated the three-dimensional coordinates of points along the intersection of the plane and the objects. The edge detection technique locates discontinuities in one plane and links them to similar discontinuities in previus planes.

They found out that the spacing between light planes makes a significant difference in the complexity of the procedure that links discontinuities from one plane to the next. When the light planes are close together relative to the size of the object features, matching is essentially easy. When the planes are far apart, the matching is extremely difficult; this effect gives a sampling rate estimate analogous to the Nyquist limit in sampling theory. A deeper sampling analysis will be done in the section 2.5. For the physical acquisition of the pictures they borrowed a one-meter-long optical track and gathered multiple images while moving a camera manually along it.

By different possibilities of camera movements on the track (e.g. straight ahead) they realized that it would be easier to make such measurements if they aimed the camera perependicularly to the track instead since the path of a scene point in the multiple views will follow a straight-line trajectory in time (whereas it will follow a hyperbolic trajectory if the camera is moving straight-ahead).

The latter can be proven using the next diagram:

Analyzing the figure 2.13 one can see that the one-dimensional images are at distance $h$ in front of the lens centers, while the feature point $p$ is at a distance $D$ from the linear track along which the camera moves right to left. By similar triangles one has

$$\begin{aligned} \Delta U = u_2 - u_1 &= \frac{h(\Delta X + X)}{D} - \frac{hX}{D} \\ &= \Delta X \frac{h}{D} \end{aligned} \tag{2.1}$$

where $\Delta X$ is the distance traveled by the camera along the line, $\Delta U$ the distance the feature moved in the image plane, $h$ is the focal distance of the camera and $D$ is the depth of the feature point. By the Equation 2.1 the change in image position is a linear

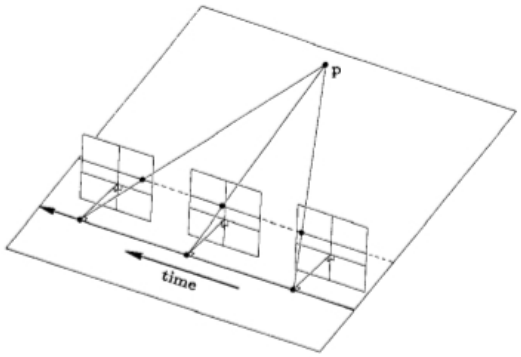



FIGURE 2.12: Lateral motion with camera perpendicular to the track, Figure taken from [26] p.9

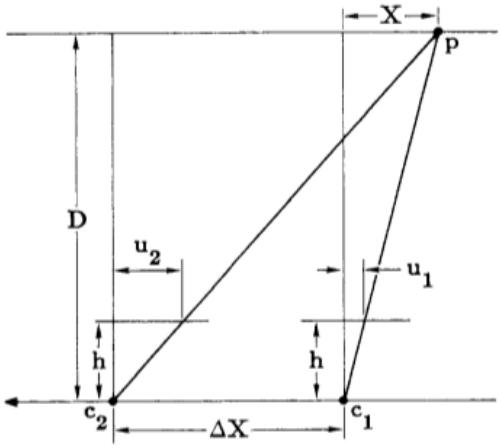



FIGURE 2.13: Lateral motion epipolar geometry, Figure taken from [26] p.9

function of the distance the camera moved; this equation can be rearranged as follows to yield a simple expression for the distance of a point in terms of the slope of its line in the EPI (concept that will be explained in more detail in the next sections):

$$D = h\frac{\Delta X}{\Delta U} \tag{2.2}$$

so if one constructs the spatio-temporal paths of feature points one can get its depth in the scene with respect to the image plane by measuring the slope of its lines with the Equation 2.2, the spatial part just considers one dimension of the image planes, this is possible without losing information since in this setup the points follow a linear paths.

We already mention words as epipolar plane, or epipolar plane image but we have not define anything yet. There are two different approachs to epipolar geometry, one is using functional analysis and permits the study of approximation errors of recovered 4D Light Fields, and the other is geometrical which permits an straight forwart implementation. In this subsection we will shortly expose them.

### 2.4.3 Functional analysis approach to EPI

At the beginning of this chapter we mentioned the parallel plane approach to 4D light field (recall Figure 2.3), the idea of epipolar geometry is based on this representation. As in Figure 2.3 let the two parallel planes be called $\pi_0$ and $\pi_1$ with coordinates $(s,t)$ and $(u,v)$ respectively. In this scheme, the 4D Light Field will be a function $L_4 : \mathbb{R}^4 \longrightarrow \mathbb{R}^3$ with the radiance $\mathbf{r} \in \mathbb{R}^3$ given as

$$\mathbf{r} = L_4(u,v,s,t)$$

If we fix one of the two coordinates on $\pi_0$, say $t$, so that $\pi_0$ reduces to a line, the ray space of the resulting light field will span the $u,v$ and $s$ dimensions of the original ray space; lets called this parameterized light field a 3D light field, and can be denoted as a function $L_3 : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$. The radiance $\mathbf{r} \in \mathbb{R}^3$ of a light ray is given then as

$$\mathbf{r} = L_3(u,v,s)$$

where $s$ is the 1D ray origin and $(u,v)$ represent the 2D ray direction. One can obtain a 2D slice of light field by fixing another parameter. A $uv$-slice fixing $s$ and $t$ is simply a perspective pinhole image $I_{s,t}(u,v)$ which is a camera with no lense but a small aperture instead. A $vs$- or $ut$- slice is known as a *push-broom image* and can be obtained using a line-sensor sweeping the scene in the direction orthogonal to its linear sensor alignment [27].

A $us$-slice is obtained by reducing (fixing) one dimension, $v$, also from $\pi_1$. This slice is commonly called *flatland light field*, it represents a light field of a hypothetical height-less world, where the light field is parameterized by two lines instead of planes.

For geometrical reasons explained in the Subsection 2.4.4 this slices are called *epipolar-plane images* (EPI) when the cameras can be represented as pinhole cameras, i.e., if one can place the image plane between the scene points and the camera center [26]. We will denote an EPI as $E_v : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$, with radiance

$$\mathbf{r} = E_v(u,s) \tag{2.3}$$

of a ray at position $(u,s)$ and fixed parameter $v$.

### 2.4.4 Geometrical Approach to EPI

Lets assume that we have two cameras modeled as pinholes with the image planes in front of the lenses, using Figure 2.14

For each point $P$ in the scene, there is a plane, called the *epipolar plane*, that passes through the point and the line joining the two lens centers. The set of all epipolar planes is the *pencil* of planes passing through the line joining the lens centers. Each epipolar plane intersects the two image planes along *epipolar lines*. All the points in an epipolar plane are projected onto one epipolar line in the first image and onto the corresponding epipolar line in the second image.

This lines are important for stereo processing since they reduce the search required to find matching points from two dimensions to one; thus, to find a match for a point along an epipolar line in one image, is just necessary to search along the corresponding epipolar line in the second image; this is equivalent to the already mentioned *epipolar constraint* for a sequence of two images. Finally an *epipole* is the intersection of an image plane with the line joining the lens centers.

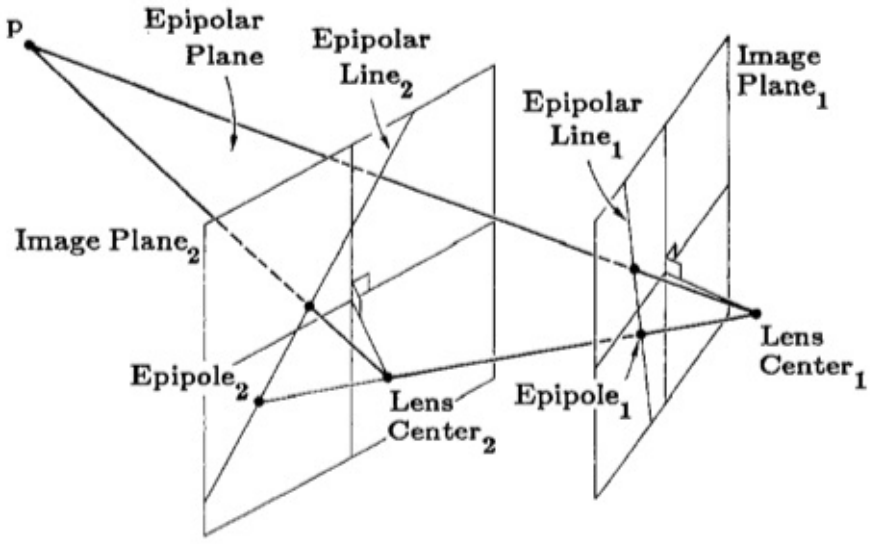



FIGURE 2.14: Stereo vision configuration, Figure taken from [26] p. 14

The epipolar constraint can be generalized for sequences of more than two images when the camera is moving in straight line and all the lenses centers are collinear, so all pairs of camera positions produce the same pencil of epipolar planes, then straight line motion of camera defines a partition of the scene into a set of planes. If the lenses centers are not in a line, the epipolar planes passing through a scene point differe in between cameras so the one-dimensional search feature will not be possible.

Since the point of an epipolar plane are projected onto one line in each image, all the info about them is contained in that sequence of lines, the image constructed from this sequence of line is called *epipolar plane image*(EPI) and contains all the information about the epipolar plane (check Figure 2.15)

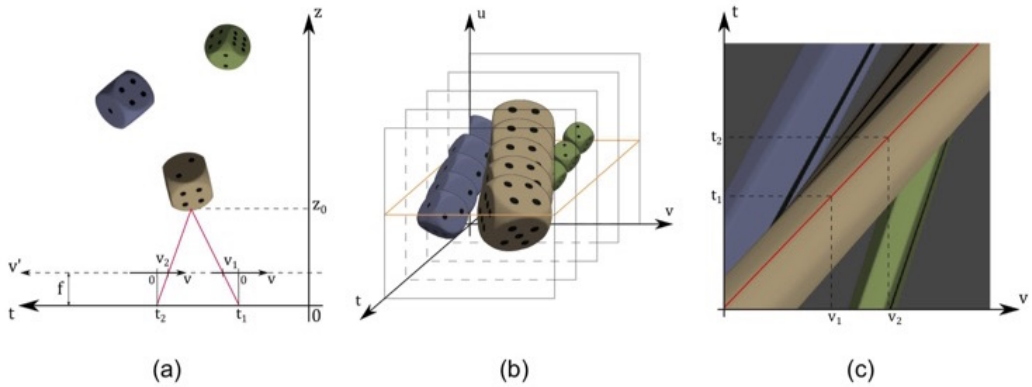



FIGURE 2.15: Epipolar plane image (EPI) formation, (a) Capturing setup, (b) Stack of captured images, (c) Example of EPI. Figure taken from [3] page 2

If one has the EPI of a sufficiently dense sequence, one can estimate then the depth of each point of the scene with the slope of the lines in the EPI using Equation 2.2 and obtain the depth map.

## 2.5 Physical and computational setup for sparse acquisition of epipolar plane

One of the downsides on the references in this topic that we found out was the lack of detailed explanation of the followed pipeline that the group or researcher in question used to take and process the set of images of a scene, to go from a sequence of raw pictures to the epipolar plane images of the sequence; in most of the papers and books one gets a black box of expensive privative computer vision software used to detect and track pictures in the sequence; in some papers is also not clear the whole reconstruction procedure in the sense that they just present the algorithm but not the implementation code which makes impossible to reproduce and improve their implementation or even to assure that it actually works.

In this thesis we are trying to make every single detail clear in order to give the reader the tools to try themselves each step of the acquisition/processing/reconstruction technique and if possible improve it.

### 2.5.1 Physical setup and sampling rate

We already saw that there are a plenty of techniques on acquire the light field of a scene, and we will use the approach of multiple views of a moving camera proposed by Bolles et al.

By lack of equipment we did not take the images but used the datasets provided by the research group of Professor Markus Gross in the Disney Research Center at Zurich used for their publications [30], [31], [32] and [33], all of them about scene reconstruction, outlier removal and motion flow applied to new filming techniques. One can find the datasets in their webpage `https://www.disneyresearch.com/project/lightfields/` with detailed description of their setting.

They provide five different datasets that are made of sequence of images named after the objects that appear in the scene: Mansion, Church, Couch, Bikes and Statue; these datasets have been widely used by the community (see [3]). In all the cases they used a digital SLR camera translated motorized linear stage to capture the multiple views (with the camera facing perpendicularly with respect of the stage). One can observe in Figure 2.16 the used stage and camera.

The reconstruction of the light field in a scene has some restrictions in the sampling rate, it is clear that successive views that are too separate from each other will make the task more difficult if not impossible. Recalling Equation 2.1 we have that

$$\Delta U = \Delta X \frac{h}{D}$$

where $\Delta X$ is the distance traveled by the camera between succesive views, $\Delta U$ is the distance the feature moved in the image plane, $h$ the focal distance (distance between the lens center and the image plane) and $D$ the distance between the stage where the camera is moving and the feature point.

Following the idea of Vagharshakyan et al. [3], by assuming a horizontal sampling rate $\Delta U$ satisfying the Nyquist sampling criterion for scene's highest texture frequency, i.e. the sampling frequency is at least the double of the scene's highest texture frequency,

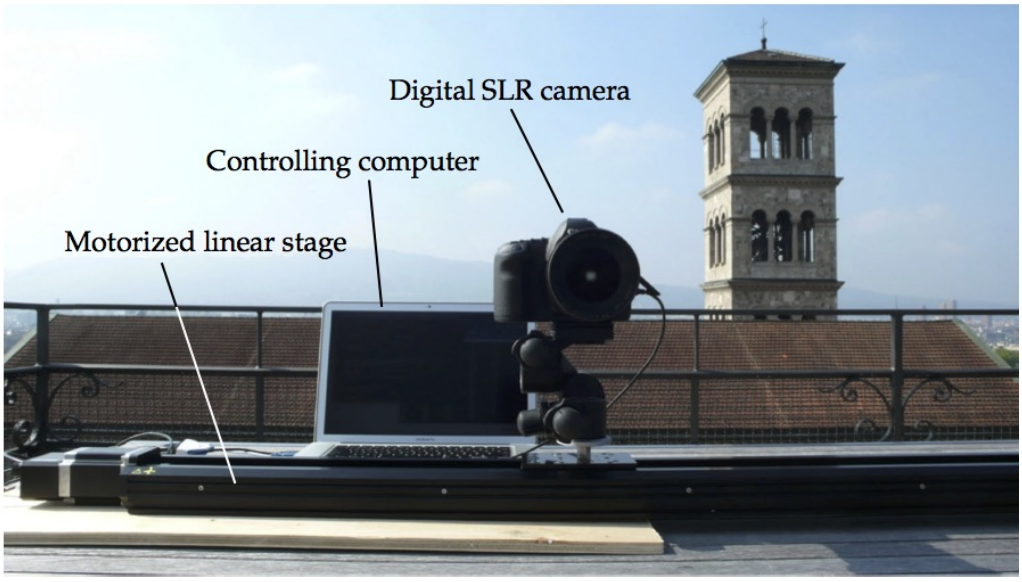



FIGURE 2.16: Acquisition setup with a digital SLR camera translated by a motorized linear stage, both controlled remotely from a computer. Figure taken from [30] p.27

one can relate the required camera motion step (sampling) with the scene depth. For given $D_{min}$ the sampling rate $\Delta X$ should be such that

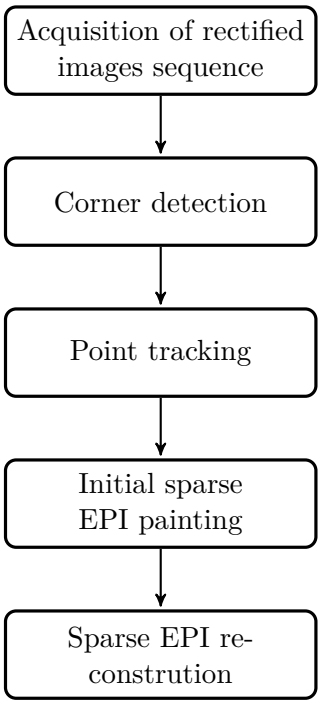
$$\Delta X \leq \frac{D_{min}}{h} \Delta U \tag{2.4}$$

in order to ensure a maximum of 1 pixel disparity between nearby views, which will avoid aliasing and other artifacts. Vagharshakyan et al. also proved that by selecting the equality for $\Delta X$ in Equation 2.4, one maximizes the baseband support, which helps in designing reconstruction filters; in particular, simple separable filters like linear interpolators can be used. The problem in this thesis is the reconstruction densely sampled EPIs (and thus the whole LF) from their decimated and aliased version produced by a higher camera step $\Delta X$ than the one in Equation 2.4 by using some sparse representation of the EPIs.

In the case of the setting used by the group of the Disney Research Center, the images were captured by using a Canon EOS 4D Mark II DSLR camera and a Canon EF 50mm f/1.4 USM lens and a Zaber T-LST1500D motorized linear stage to drive the camera to the shooting positions; an image of used setup can be found on Figure 2.16. The camera focal length was 50 mm and the sensor size was $36 \times 24$mm, PTLens was used to radially undistort the captured images, and Voodoo Camera Tracker was used to estimate the camera poses for rectifiction (is very important that the images are rectified to be able to track points); by its number of corners (features easy to track) the **Church** data set which consists in 101 pictures was the one studied in this thesis, here the camera separation is $\Delta X = 10mm$ which attains the Nyquist sampling bound mentioned in Equation 2.4.

### 2.5.2 Followed pipeline

The next diagram shows the general followed pipeline from acquisition to LF reconstruction:




The last part of diagram can be represented diagramatically by Figure 2.17

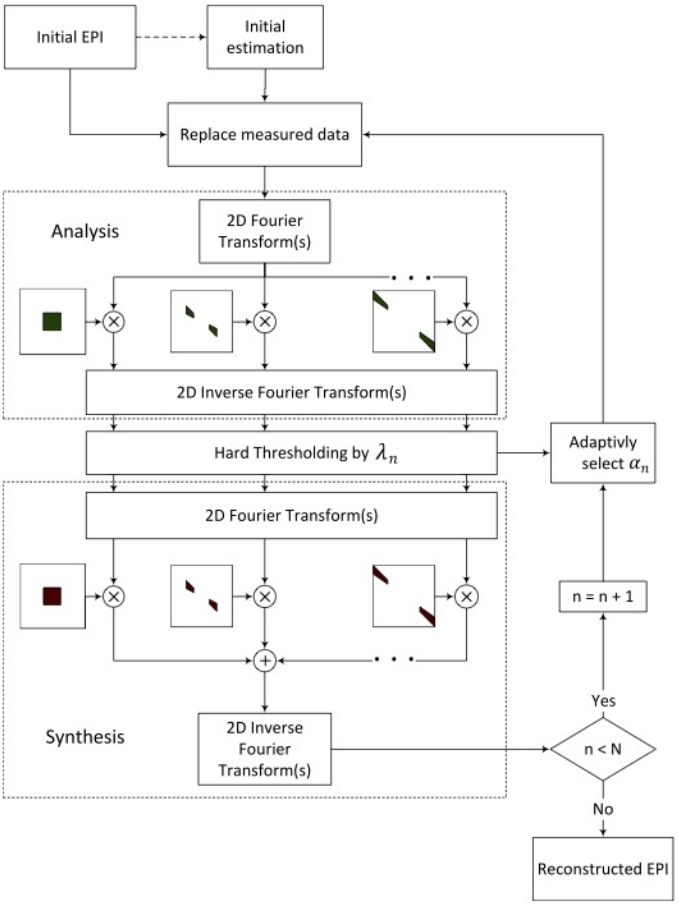



FIGURE 2.17: Diagram of sparse EPI reconstruction algorithm, Figure taken from [3] p. 7

The first step in the pipeline (Acquisition of rectified images sequence) was already explained in Subsection 2.5; the three middle steps (corner detection, point tracking and initial sparse EPI painting) will be explained in the last subsections of this chapter. Finally the last step (sparse EPI reconstructin) will be described in detail in Chapter 4.

### 2.5.3 Geometric construction of epipolar lines

To have closer in mind, in stereo vision when we have two different points of views of a scene (that can be interpreted as two different cameras pointing to the same scene) a line that passes through the lens center of one camera maps to a point $x$ in the image plane and to a line in the image plane of the other camera, this line is called epipolar line, see Figure 2.18

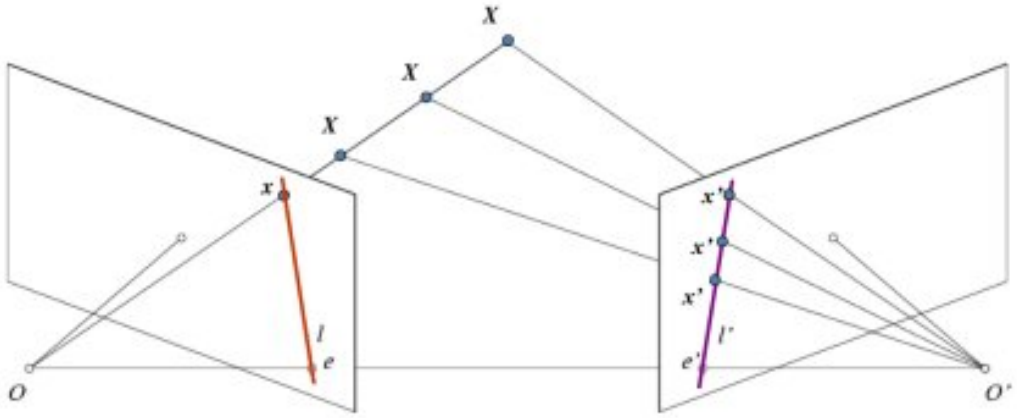



FIGURE 2.18: Epipolar line correspondent to a scene point $X$. Figure taken from `http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html`

As we mentioned, every point on the line $OX$ projects to the same point on the left image plane, this implies that just with one image we cannot triangulate the 3D point on the scene. If the points $x$ and $x'$ (corresponding to the same scene point) on the two image planes are known the proyection lines ($Ox$ and $O'x'$) most intersect exactly at X, so the coordinates of points $X$ on the scene can be calculated from the coordinates of the two image points; this means that with two perspectives is possible to triangulate 3D points. Occlusion causes problems since some important features can be blocked by other objects and this can be solved by taking more pictures. The epipolar geometry is based in this result.

Let $l'$ be the epipolar line in the right image plane correspondint to $X$, this is also the projection of the line $OX$ on this image plane, by epipolar constraint to find the matching point in the right image one needs just to search in the epipolar line correspondent to $X$, this allows us to have a better performance and accuracy in feature tracking algorithms. The plane $XOO'$ is called *epoipolar plane*. All the epipolar lines at each image intersect in one point called the epipole (in the Figure 2.18 the epipoles correspond to the points $e$ and $e'$) and every epipolar plane pass throught the epipoles; one can also find the epipoles with the intersections of the line that joins the lens centers $O$ and $O'$ and the image planes.

To be able to construct algorithmically the epipolar lines we used the method implemented in the famous computer vision toolbox OpenCV (`http://opencv.org/`),

where one can use the concepts of **Fundamental Matrix (F)** and **Essential Matrox (E)**; this matrices include all the realtive spatial information of one of the image planes with respect to the other (rotation and translation), see Figure 2.19

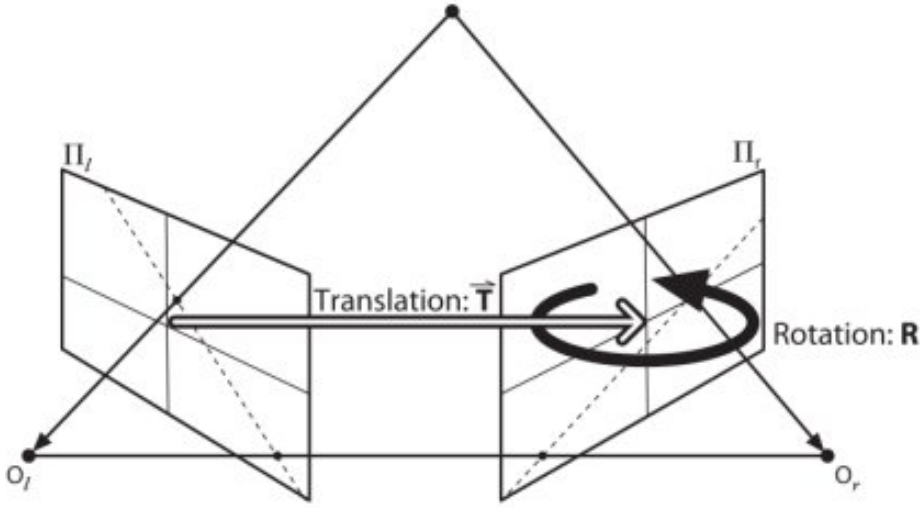



FIGURE 2.19: Essential Matrix. Figure taken from `http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html`

Lets define and construct precisely both matrices:

- **Essential Matrix (E):** It contains the information about rotation and translation of the image plane, which decribes the location of the second camera relative to the first in global coordinates (i.e. euclidean spatial coordinates of the 3D scene). To construct it lets pick one coordinate system to work in and do our calculations from there, for instance lets coose our coordinates centered on $O_l$ (left camera's center), in this coordinates the location of the observed point $P$ is $P_l$ and the origin of the other camera is at $T$. The location of $P$ as seen by the right camera is $P_r$ in our coordinate system, where

$$P_r = R(P_l - T) \tag{2.5}$$

with $R$ the associated rotation matrix, to relate this we need to introduce the epipolar plane. The equation of a plane which passes trough a point $a$ with normal vector $n$ is $(x - a) \cdot n = 0$, in this case the coordinates of the point $P_l$ which is in the epipolar plane will be

$$(P_l - T)^\intercal (T \times P_l) = 0 \tag{2.6}$$

combining Eq. 2.5 and Eq. 2.6 we obtain then that $(P_l - T) = R^{-1}P_r$, but rotation matrix are orthogonal so $R^\intercal = R^{-1}$ then $(R^\intercal P_r)^\intercal (T \times P_l) = 0$, one can define then the matrix $S$ such that $T \times P_l = SP_l$ so

$$S = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

this imples that $(P_r)^\intercal RSP_l = 0$. One defines $E = (P_r)^\intercal EP_l$ (where $E$ is the essential matrix), now to get back to global coordinates, one uses the projection equations $p_l = f_l P_l/Z_l$ and $p_r = f_r P_r/Z_r$ where $f_l$ and $f_r$ are the focus length of each camera and $Z_r$,$Z_l$ are the $Z$ coordinate fo the point with respect of each camera; divinding them by $Z_l >_r /f_l f_r$ one obtains the equation for the epipolar line:

$$p_r^\intercal E p_l = 0 \tag{2.7}$$

since the essential matrix $E$ is a rank deficient matrix (i.e. if $E$ is of size $n \times n$ there are fewer $n$ nonzero eigenvalues) the Equation 2.7 is the equation for a line, but we are interested in camera coordinates (pixel coordinates) and $E$ does not relates them, rather relates global coordinates, even though one can us $E$ to construct the fundamentla matrix $F$ that will do the work.

- **Fundamental Matrix (F):** It contains the same information as $E$ in addition to information about the intrinsics of the cameras (pixel coordinates). If $p$ is a point and $M$ is the camera intrinsic matrix (which projects the image to the pixels), then $q = Mp$ is a point in the camera's coordinates, using this and the Equation 2.7 one has

$$q_r^\intercal (M_r^{-1})^\intercal E M_l^{-1} q_l = 0 \tag{2.8}$$

so one defines the fundamental matrix $F$ as $F = (M_r^{-1})^\intercal E M_l^{-1}$ so that $q_r^\intercal F q_l = 0$, then $F$ is just like $E$ but $F$ operating in the image pixel coordinates rather than in the physical coordinates.

It is clear that finding the epipolar lines does not require complicated mathematical concepts just linear algebra and classical geometry, for a more detailed explanation of the fundamental and essential matrices and its implementation in OpenCV we recommend to read the chapter 12 of [28] which is strongly based on the more theoretical book "Multiple View Geometry in Computer Vision" by R. Hartley and A. Zisserman [29]. We are assuming here that we have a form to find matching points in between the images, but this is in our experience the hardest task on the EPI construction, and there are different ways to tackle which we will explain in the next subsection.

### 2.5.4 Tracking point algorithms

Tracking a point in a sequence of images of the same scene is a very common task in computer vision; it can be applied to analyse motion flow in a video in order to predict position of an object in future frames. The task consists mainly in two parts: First, you need to detect feature points that are easy to track (e.g. corners) and second you need to follow them in the different frames. In this subsection we will present first some feature detection algorithms that are used commonly in motion flow tracking with the advantages and disadvantages of each one.

- **SIFT (Scale Invariant Feature Transform):** As its name suggests it SIFT is a feature detector that is scale invariant. In the universe of computer vision related algorithms there exist plenty of image feature detection algorithms; some of them are corner detectors which are rotation invariant (e.g. Harris and Shi Tomasi), i.e. even if the image is rotated we can find the same corners; this makes a lot of sense sicne corners remain corners even if the image is rotated, but they are not necessarily scaling invariant, for example a corner in a small image within a small window is flat when is zoom in with the same window, see Figure 2.20

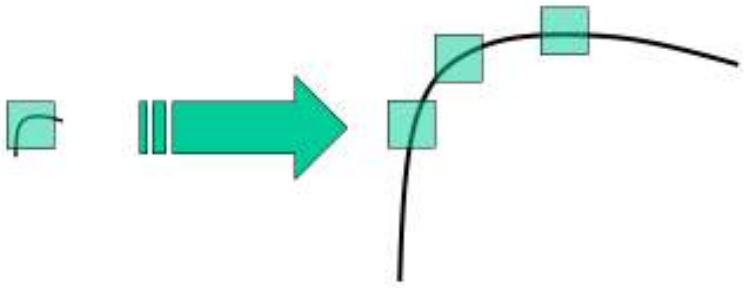


FIGURE 2.20: Scaling a corner with constant window size does not output a corner. Figure taken from `http://docs.opencv.org/trunk/da/df5/tutorial_py_sift_intro.html`

In 2004, D. Lowe of University of British Columbia published the paper "Distintive Image Features from Scale-Invariant Key Points" [34] where he presented this scaling-invariant feature detector that is known as the first of its kind and state of the art (OpenCV contains an implementation of the algorithm just in the developers version of the API).

The broad idea of the algorithm is as follows: From the Figure 2.20 is obvious that to detect windows with different scale. It behaves correctly with small corners, but to detect large corners we need larger windows. For this, scale-space filtering is used; Laplacian of Gaussian (refering to the famous blob detector spatial filter nicely explained in [35]) is found for the image with various standard deviations $\sigma$ (which controls the scales); this acts as a blob detector for blobs of different sizes. One finds the local maxima accross the space and scale which give us a list of $(x, y, \sigma)$ values (see Figure 2.21) which means there is a potential key point $(x, y)$ at scale $\sigma$

To draw the epipolar lines of a pair of images we can use SIFT as the feature points matching algorithm but is not very useful when trying to track the same points in a lot of successive images for two main reasons; first is very costly computationally since it needs to detect corners several times for different sizes and second, in the practice when we tried to implement it with OpenCV it was not mantaining the order of the corners so when we have too many corners there was not a straight-forward way to keep track along the sequence of pictures.

- **Harris Corner Detector:** This corner detector was introduced by Chris Harry and Mike Stephens in the 1998 paper "A combined corner and edge detector" [36], and their idea was very simple. This algorithm basically finds the difference in intensity for a displacement of $(u, v)$ in all directions. This is expressed as bellow:

$$E(u,v) = \sum_{x,y} w(x,y)(I(x+u, y+v) - I(x,y))^2 \tag{2.9}$$

where $w$ is a windows function (e.g. rectangular or gaussian), and $I(x, y)$ is the intensity of the image at the point $(x, y)$. For corner detection one has to maximize the functional $E(u, v)$, applying Taylor expansion until second order one gets the quadratic equation

$$E(u,v) \approx \begin{pmatrix} u & v \end{pmatrix} M \begin{pmatrix} u \\ v \end{pmatrix}$$

FIGURE 2.21: OpenCV implementation of SIFT algorithm that detects corners of different sizes. Figure taken from `http://docs.opencv.org/trunk/da/df5/tutorial_py_sift_intro.html`

where

$$M = \sum_{x,y} w(x,y) \begin{pmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{pmatrix}$$

where $I_x$ and $I_y$ are directional derivatives of the intensity. The main part comes when after this they created a score, this scores will indicate if a window can contain a corener or no and is given by the following relation

$$R = det(M) - K(\text{tr}\,(M))^2 = \lambda_1 \lambda_2 - K(\lambda_1 + \lambda_2)^2 \tag{2.10}$$

where $\lambda_1$ and $\lambda_2$ are the eigenvalues of $M$. The criterion with the score has the next cases:

1. If $|R|$ is small, i.e. $\lambda_1, \lambda_2$ are small, the region is flat.
2. If $R < 0$, i.e. $\lambda_1 \gg \lambda_2$ or viceversa, the region is an edge.
3. If $R$ is large, $\lambda_1$ and $\lambda_2$ are large and $\lambda_1 \sim \lambda_2$, then region is a corner.

for a graphical representation of this conditions see Figure 2.22.

OpenCV offers a faithful implementation of Harris Corner Detector, but we rather used a modification of this algorithm that works better, the so called Shi-Tomasi Corner Detector.

- **Shi-Tomasi corner detector:** After Harris and Stephens proposed their corner detector in 1994 J. Shi and C. Tomasi proposed a variation on their paper "Good Features to Track" [37] which shows better results compared with Harris work.

  Shi-Tomasi changes the scoring function that gave criteria for corner detection in Harris (see Equation 2.10) to the form

$$R = \min(\lambda_1, \lambda_2) \tag{2.11}$$

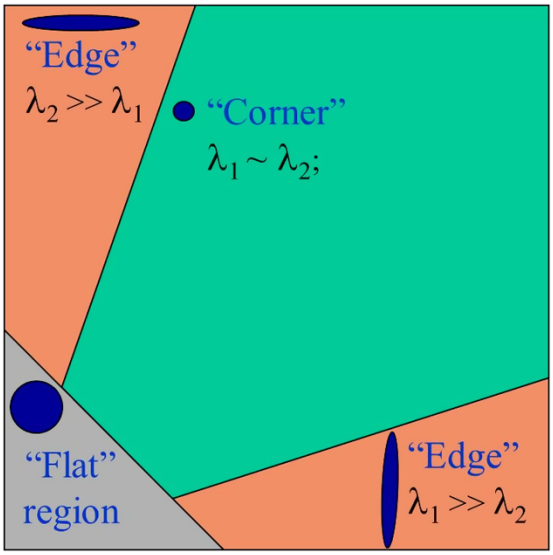



FIGURE 2.22: Diagram representing the criterion of corner detection for Harris detector, the axis $x$ represents $\lambda_1$ and axis $y$ represents $\lambda_2$

as in the case of the Harris Corner Detector, if $|R|$ is greater than a threshold value $\lambda_{\min}$, it is considered as a corner. The $\lambda_1$vs.$\lambda_2$ space will now look as in Figure 2.23

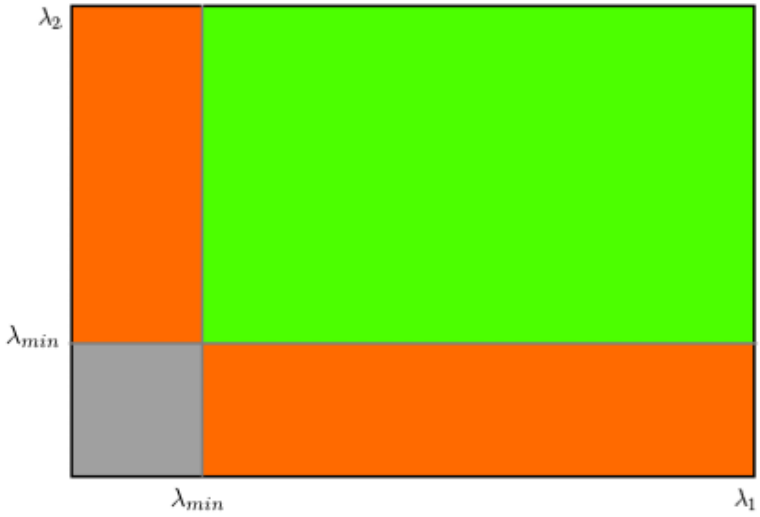



FIGURE 2.23: $\lambda_1$vs.$\lambda_2$ space for Shi-Tomasi corner detector, as in Harris detector's case, the upper right area corresponds to corners, the upper left and lower right correspond to edges

There is also an straight forward implementation of this algorithm on OpenCV, we used this algorithm to find the $N$ strongest corners and then track them with Lucas-Kanade algorithm. Even the Shi-Tomasi corner detector is rotation invariant since the trace and the determinant of the matrix $M$ are rotation invariant they are not scaling invariant like the SIFT algorithm which is slower than the former. In the case of the sequence that we are working with the images were taken very close to each other so the scale of the features wont change significantly.

We already explained the different options for feature detection algorithms and

that the option picked was the Shi-Tomasi corner detector to track the strong corners obtained by the Shi-Tomasi algorithm we used the Lucas-Kanade method explained as follows:

- **Lucas-Kanade method:** We would like to associate a movement vector $(u, v)$ to every such "interesting pixel" (strong corner point) in the scene obtained by comparing two successive images with the next two assumptions:

  1. The two images are separated by a small increment $\Delta t$, such that the objects have not displaces significantly (the algorithm works best with slow moving objects).
  2. The images depict a natural scene containing textured objects exhibiting shades of gray (different intensity levels) which change smoothly.
  3. The pixel intensity of an object does not change in consecutive frames.
  4. Neighbouring pixels have similar motion.

  With this assumptions consider a pair in $(x, y)$ at time $t$ with intensity $I(x, y, t)$, it moves by distance $(dx, dy)$ in next frame taken after $dt$ time. Since those pixels are the same and intensity does not change we can say

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

  Expanding with Taylor the right hand side we obtain the following equation

$$I_x u + I_y v + I_t = 0 \tag{2.12}$$

  where

$$I_x = \frac{\partial I}{\partial x} \quad ; I_y = \frac{\partial I}{\partial y}$$
$$I_t = \frac{\partial I}{\partial t}$$
$$u = \frac{\partial x}{\partial t} \quad ; v = \frac{\partial y}{\partial t}$$

  The Equation 2.12 is known as the **"Optical Flow Equation"**.Computing the gradient of the intensity we obtain $(I_x, I_y, I_t)$, and we aim to know the flow by solving the equation for $(u, v)$. In 1981, Bruce D. Lucas and Takeo Kanade proposed a method to solve this in their paper "An iterative image registration technique with an application to stereo vision" [38]; the made the assumptions that we already mentioned before.

  By assumption neighbouring pixels have similar motion, lets take a $3 \times 3$ patch around the point (in our case corner), then all the nine points of the patch have the same motion. We can find $(I_x, I_y, I_t)$ for this nine points; thus the problem reduces to solve nine equations with 2 unknowns variables which is over-determined. As is explained in detail on [38] a better solution is obtained with least squares fit method. In this setting the problem has the solution

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \sum_i I_{x_i}^2 & \sum_i I_{x_i} I_{y_i} \\ \sum_i I_{x_i} I_{y_i} & \sum_i I_{y_i}^2 \end{pmatrix}^{-1} \begin{pmatrix} -\sum_i I_{x_i} I_{t_i} \\ -\sum_i I_{y_i} I_{t_i} \end{pmatrix} \tag{2.13}$$

  obtaining with this the vector flow of the features in the scene. By its simplicity we used the OpenCV implementation of this algorithm to track the $N$ strongest

corners (found by Shi-Tomasi corner detector) in the image sequence **Church**. In the next subsection we will show explicitly how did we implemented the Lucas-Kanade method with the Shi-Tomasi algorithm to detect and track strong corners in our dataset, including the code in python. We will also show how to paint the Epipolar plane based on the results of this procedure.

### 2.5.5 Procedure for tracking and painting the EPIs

The maximum number of strong corners obtained by the Shi-Tomsi in the first view was 336, one can see in Figure 2.24 the first image of the curch and in Figure 2.25 the 336 strong corners found by the Shi-Tomasi detector; one can also see in Figure 2.26 the last image (number 101) of the church and in Figure 2.27 the final position of the points correspondent to the tracked corners. Finally, in the Figure 2.28 one can see the path of the corners tracked through the sequence of images in the data set. The code used to detect the $N$ strongest corners with Shi-Tomasi detector and track them through the sequence of images in the Church data set using Lucas-Kanade method is presented in Appendix A. Using this code the time elapsed to detect and track 336 corners along 101 pictures in the Church data set sequence was **16.36 seconds** in a Macbook Pro with OSX 10.10.5, with 8GB memory, 2.7 GHz Intel Core i5 processor and Graphic Card Intel Iris Graphics 6100 1536 MB.

FIGURE 2.24: First image of the church data set

We are trying to construct the epipolar plane images correspondent to the sequence of different views of the church; for that we will follow the method that Bolles proposed in [26]. We will use that the points follow a straight line trajectory along the sequence due that the camera followed an straight line so each point in the first image will move in its correspondent epipolar line which will be parallel to the $x$-axis, since the camera points orthogonally with respect to the to the scene.

FIGURE 2.25: 336 corners found in the first image of the church with different colors corresponding to different features

FIGURE 2.26: Last image of the church data set

For each strip $y_0 - \epsilon \leq y \leq y_0 + \epsilon$ that is parallel to the $x$-axis in the initial image we ploted for points correspondent to different features the $x$ entry (which corresponds to the $u$ entry in the two-planes 4D light field model) with respect to the time (the sequence of images).

FIGURE 2.27: Last position of corners in the last image of the church

FIGURE 2.28: Path of points tracked in the image sequence of the church, one can observe that the trajectories of the features are more or less straight lines, with some nuerical and algorithmical errors that can be ignored. The image is presented in shades of gray since the Lucas-Kanade method can be implemented with good performance in shades of gray

Since in comparsion with the actual resolution of the pictures ($1024\text{px} \times 683\text{px} = 699392\text{px}^2$) the number of corners that we could detect was very small (about 0.04% of the total number of points) taking strips of points with constant $y = y_0$ that are very tight will not capture a lot of tracked points; the distribution of the points as one can see in Figure 2.25 is not homogeneous at all.

In order to have a trustworthy light field reconstruction we took Epipolar plane images corresponding to $y$-strips with different thickness depending on the density of tracked points for the corresponding $y$, for example more tight at the bottom part of the pictures where we have a lot of tracked points due the bushes and trees and broader at the top where there is not a lot of tracked points due the almost homogeneous sky.

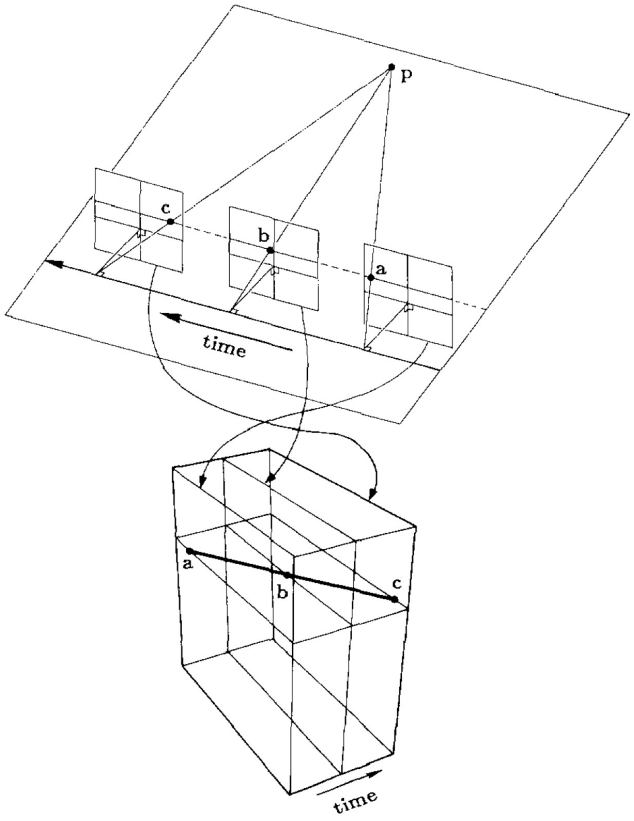



FIGURE 2.29: Feature point tracking for lateral camera motion. Figure taken from [26] p. 16

We can see in Figure 2.32 a strip at the bottom of the image, centered at $y_0 = 673$, with width $2\epsilon = 20$ that captures 48 different tracked points corresponding to 8 different features, the dense EPI associated with this strip is in Figure 2.33 and its sparse form obtained by measuring each 4 rows at the EPI is in Figure 2.34. We will present a deeper analysis on the sparse measure and the main results of Shearlet-based inpainting an sparse EPI and measuring the depth map which is the main topic of this thesis at Chapter 4.

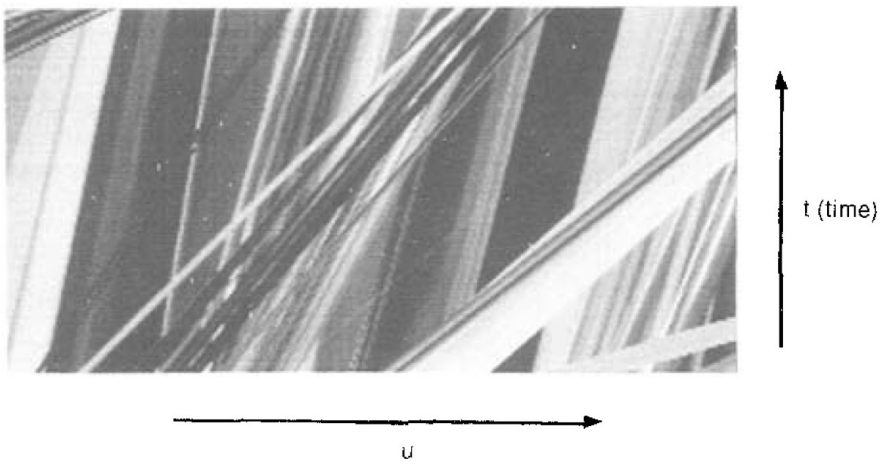



FIGURE 2.30: EPI correspondent to some strip in the sequence of images, we are looking to get the EPI for the Church data set. Figure taken from [26] p. 17

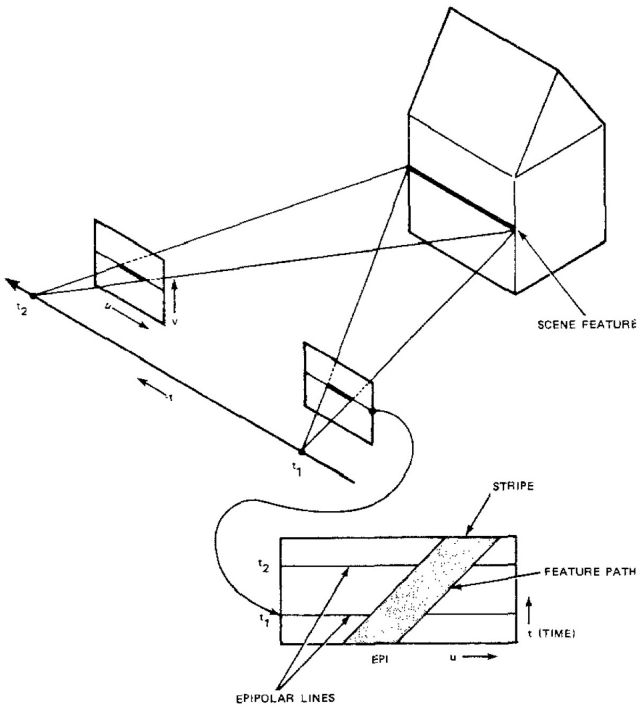



FIGURE 2.31: Horizontal line of a feature (house) corresponds to a diagonal strip its correspondent EPI. Figure taken from [26] p. 17

FIGURE 2.32: Tracked points on a strip centered at $y_0 = 673$ with width $2\epsilon = 20$

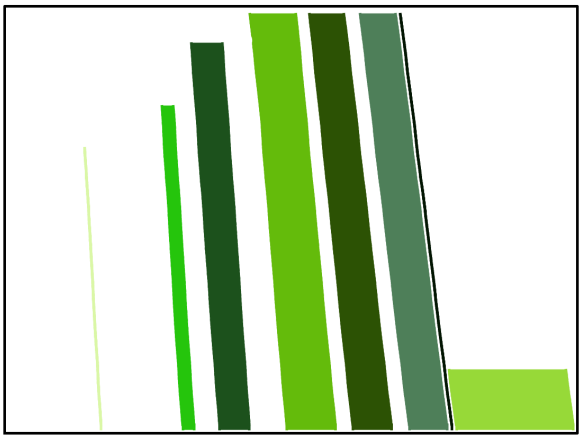


FIGURE 2.33: Dense Epipolar plane image associated with the strip on Figure 2.32, the horizontal axis is the spatial coordinate $x$ and the vertical axis is the time

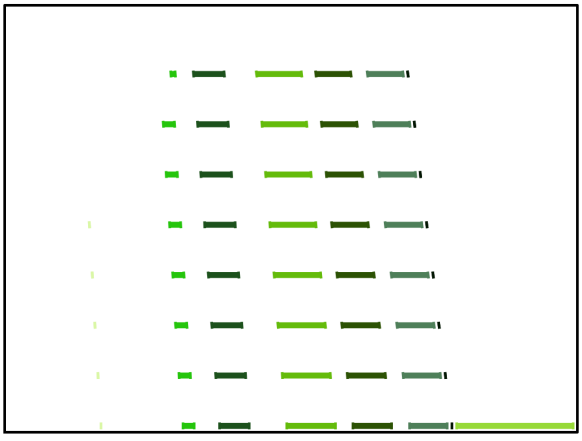


FIGURE 2.34: Sparse Epipolar plane image associated with the strip on Figure 2.34 by measuring each 4th row on Figure 2.33

# Chapter 3

# Shearlets

We want to be able to express efficiently Epipolar Plane Images, since this will reduce the number of minimum views needed to recover the light field of a scene; this task can be achieved by understanding the EPIs as signals and using signal processing machinery developed in the last twenty years, in this chapter we will explain in detail state-of-the-art methods on signal sparse-representation.

One can think a signal as a function (or something that can be represented as) that contains information about the behavior or attributes of some phenomenon [39], by this definition it actually could be a lot of things; it also depends the area you are working on, this definition will work or not. For example, in signal processing, arbitrary binary data streams are not considered as signals. For the sake of simplicity in this thesis we will agree to define a signal as a function that could represent video, image or audio and it will be either analog (evaluated with continuous parameters) or digital (evaluated with discrete parameters).

In signal processing and applied harmonic analysis one can generally represent the signals in a space-time domain, but one cannot get always meaningful information in this representation; plenty of different signal transforms have been proposed along time, this transforms are obtained generally by finding basis of certain functional spaces (e.g. $L^2(\mathbb{R}^2)$) and present different features like sparse representation of signals that permits an efficient processing and storing of them. The most known signal transform for its effectiveness and tradition is the Fourier Transform, proposed by the French Mathematician Joseph Fourier, on his paper "Théorie analytique de la chaleur" in 1822 where he showed that some functiones could be written as an infinite sum of sines and cosines.

If $f \in L^2(\mathbb{R}^n)$ then its Fourier transform $\hat{f}$ will be

$$\hat{f}(\xi) := \int_{\mathbb{R}^n} f(x) e^{-i\langle x,\xi\rangle} dx$$

one can think of the coordinates $\xi$ of the Fourier space as frequencies of $f$, as one can see the Fourier transform $\hat{f}$ just gives information about the frequencies contained in $f$, but not at which time they occur; moreover, small changes in the neighborhood of some point $x \in \mathbb{R}^n$ could change significantly its Fourier transform, in general one would like a transform that is robust under this kind of changes. A direct solution for this issue is reflected in the short-time Fourier transform; whose mechanism is based in localization of the Fourier transform to a certain window of $f$ and then move the window through the whole domain. Let $g \in L^2(\mathbb{R})$ the window function, the short-time Fourier transform of $f \in L^2(\mathbb{R})$ associated with the window $g$ will be

$$S_g f(t,\xi) = \int_{\mathbb{R}} f(x)\overline{g(x-t)}e^{-ix\xi}dx = \langle f, M_\xi T_t g\rangle = (\widehat{f \cdot T_t \overline{g}})(\xi),\ t,\xi \in \mathbb{R}$$

where $T_t : L^2(\mathbb{R}) \longrightarrow L^2(\mathbb{R})$ is the translation operator with parameter $t$, given by

$$T_t f(x) = f(x-t)$$

and $M_\xi : L^2(\mathbb{R}) \longrightarrow L^2(\mathbb{R})$ is the modulation operator, given by

$$M_\xi f(x) = e^{i\xi x} f(x)$$

One can associate to this transform the atoms $\{M_\xi T_t g\}_{(t,\xi)\in\mathbb{R}^2}$; for computational purposes one can discretize the transform by taking $(t,\xi) \in \Lambda = a\mathbb{Z} \times b\mathbb{Z}$ for some $a, b > 0$; the resulting atoms $G(g,a,b) := \{g_{am,bn} = M_{bn}T_{am}\}_{(m,n)\in\mathbb{Z}}$, for some cases of $(a,b) \in \mathbb{R}$ $G(g,a,b)$ is a generating set of $L^2(\mathbb{R})$ with an explicit recovery formula and other features, such sequence of function are known as **frames** (in this particular case **Gabor frames**) and can be understand as the generalization of orthonormal bases, we will study them in detail on the Section 3.2, for a further reading about Gabor frames one can check [41].

We introduced Gabor frames to overcome some limitations of the Fourier transform; Gabor frames also present some limitations

- When the Gabor frame is also a orthonormal basis it doesn't have a good time-frequency localization [41].
- The size of the window $g$ does not change therefotherefotherefotherefotherefotherefotherefotherefotherefotherefotherefore Gabor frames are not sensible to very localized information, so for instance they will never detect a singularity or regularity information of a function.

both limitations above can be overcome using **wavelets**.

The concept of wavelets and the signal transform related was introduced first time in 1980s by the french mathematicians Morlet and Grossmann to refer to "small wave" (or *ondelette* in french) when they were studying siesimic waves (check the original paper [42]). The *continuous wavelet transform* of a function $f \in L^2(\mathbb{R})$ associated to a mother function $\psi \in L^2(\mathbb{R})$ is defined by

$$\begin{aligned}\mathcal{W}_\psi f(a,b) &= \int_{\mathbb{R}} f(t) a^{-\frac{1}{2}} \overline{\psi\left(\frac{t-b}{a}\right)} dt \\ &= \langle f, T_b D_a \psi\rangle = (f * D_a \overline{\psi}^*)(b),\ (a,b) \in \mathbb{R}^+ \times \mathbb{R}\end{aligned}$$

where $D_a : L^2(\mathbb{R} \longrightarrow L^2(\mathbb{R})$ is the dilation operator given by $D_a f(t) = a^{-\frac{1}{2}} f\left(\frac{t}{a}\right)$, and $f^*(t) = f(-t)$, $a$ is the scaling paramter (controls the size of the window) and $b$ is the translation parameter. If the mother function $\psi$ satisfy the admissibility condition

$$C_\psi := \int_0^\infty \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi < \infty$$

we will say that $\psi$ is a admissible wavelet. If one has an admissible wavelet, one can get an straightforward inversion or recovery formula as

$$f = \frac{1}{C_\psi} \int_{\mathbb{R}} \int_0^\infty W_\psi f(a,b) T_b D_a \psi \frac{da}{a^2} db$$

The sequence of wavelet atoms will be $\{\psi_{a,b}(t) = a^{-\frac{1}{2}} \overline{\psi\left(\frac{t-b}{a}\right)}\}_{(a,b)\in\mathbb{R}^+}$, so one can write the wavelet transform of $f$ as $W_\psi f(a,b) = \langle f, \psi_{a,b} \rangle$. One can discretize the wavelet transforms as by the inner product with the discrete set of wavelet atoms

$$\psi_{j,m}(t) := a^{-\frac{1}{2}} \psi(a^{-j}t - bm),\ (j,m) \in \mathbb{Z}^2,\ t \in \mathbb{R},\ (a,b) \in \mathbb{R}^+ \times \mathbb{R}$$

the set of discrete set of wavelet atoms is referred as wavelet system. Wavelets are very relevant in Signal Processing due their great features

- One can get information about the regularity of a function $f$ by estimating bounds of its wavelet transform.
- The scaling parameter permits us to detect very localized information, in particular is very effective detecting one dimensional singularities, this property leads to the construction of a Multiresolution Analysis (MRA) which is an important area in applied harmonic analysis(check [43] p. 264).
- The unified treatment of both digital and continuous transforms permits an easy computational implementation.
- It can represent sparsely one dimensional signals, in the sense that not a lot of coefficients will be significant so one can delete them and still get a good approximation of the original signal.

Over all the features that we just mentioned the one that gave most of its fame to the wavelet transform is the last one, i.e. sparse representation of one dimensional signals, for instance this porperty of wavelets is what the image compression standard JPEG 2000 is based on. It is worth it to study in more detail sparse representation of data.

It is not surprising that compression of data takes an important place in the academic research and industrial agenda nowadays. Our society generates and acquire a lot of data everyday that comes in a lot of different types and dimensions; the complexity of the processing of this raw data to extract some useful features in an understandable language grows with the dimensionality and size of the data. Even though, almost all data found in practical applications has the property that the relevant information which needs to be extracted or identified is sparse, that is, data are typically highly correlated and the essential information lives in lower dimensional subspaces (or manifolds). This information can be then captured using just few terms in an appropriate dictionary (e.g. some frame or orthonormal basis).

The sparse representation property of data is important not only for data storage and transmission but also for feature extraction, classification, and other high-level tasks; finding a dictionary which sparsely represents a certain data class involves deep understanding of its dominant properties, which are typically associated with their geometric properties; for a deep treatment of this one can read [44] and [45].

So far we have just mentioned the sparse representation property for one-dimensional signals and also the existence of straight forward and fast algorithmic implementations; the latter is based in the general machinery to construct orthonormal wavelet bases known as *Multiresolution Analysis* (MRA). In the one dimensional case, this is defined as a sequence of closed subspaces $(V_j)_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R})$ known as the scaling spaces which satisfies the following properties

(a) $\{0\} \subseteq \ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots L^2(\mathbb{R})$.

(b) $\cap_{j\in\mathbb{Z}} V_j = \{0\}$ and $\overline{\cup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R})$.

(c) $f \in V_j$ if and only if $D_2^{-1} f \in V_{j+1}$.

(d) There exists a $\varphi \in L^2(\mathbb{R})$, called *scaling function*, such that $\{T_m\varphi : m \in \mathbb{Z}\}$ is an orthonormal basis for $V_0$.

This enables the decomposition of functions into different "resolution" levels associated with the so called wavelet spaces $W_j$, $j \in \mathbb{Z}$ which are defined by considering the orthogonal complements

$$W_j := V_{j+1} \ominus V_j,\ j \in \mathbb{Z}$$

This Multiresolution Analysis let us not only to decompose $L^2(\mathbb{R})$ as a direct sum of wavelet spaces but also gives us an alternative orthonormal basis with both the wavelet and the scaling fuction, of the form

$$\{\varphi_m = T_m\varphi = \varphi(\cdot - m) : m \in \mathbb{Z}\} \cup \{\psi_{j,m} : j \geq 0, m \in \mathbb{Z}\}$$

where the scaling function take care of the low-frequency region $V_0$ and the wavelet terms of the complementary space $L^2(\mathbb{R}) \ominus V_0$. One can read [43].

In this thesis we are interested in image processing, if one would like to apply wavelets to imaging science an extension of the theory to $L^2(\mathbb{R}^2)$ is necessary. For a painless extension we can introduce the concept of tensor products of Hilbert spaces. If $\mathcal{H}_1$ and $\mathcal{H}_2$ are two Hilbert spaces the tensor product is a bilinear operator $\otimes : \mathcal{H}_1 \times \mathcal{H}_2 \longrightarrow \mathcal{H}_1 \otimes \mathcal{H}_2$ where $\mathcal{H}_1 \otimes \mathcal{H}_2$ is a new Hilbert space.

We can use strongly the fact that the tensor product of orthonormal bases is an orthonormal basis of $\mathcal{H}_1 \otimes \mathcal{H}_2$. In the case of $\mathcal{H}_1 = \mathcal{H}_2 = L^2(\mathbb{R})$ and $f, g \in L^2(\mathcal{R})$, then $f \otimes g \in L^2(\mathbb{R}^2)$ is given by

$$(f \otimes g)(x_1, x_2) = f(x_1)g(x_2),\ (x_1, x_2) \in \mathbb{R}^2,$$

and $\mathcal{H}_1 \otimes \mathcal{H}_2 = L^2(\mathbb{R}^2)$. This concepts leads to the next theorem.

**Theorem 3.1** (Two-dimensional wavelets)**.** Let $(V_j)_{j\in\mathbb{Z}}$ be an MRA for $L^2(\mathbb{R})$ with scaling function $\varphi \in L^2(\mathbb{R})$ and associated wavelet $\psi \in L^2(\mathbb{R})$. For $(x_1, x_2) \in \mathbb{R}^2$, we define

$$\begin{aligned}\psi^1(x_1, x_2) &:= \varphi(x_1)\psi(x_2),\\ \psi^2(x_1, x_2) &:= \psi(x_1)\varphi(x_2),\\ \psi^3(x_1, x_2) &:= \psi(x_1)\psi_(x_2)\end{aligned}$$

Then

$$\{\psi^k_{j,m}(x_1, x_2) = 2^{-j}\psi^k(2^{-j}x_1 - m_1, 2^{-j}x_2 - m_2):\ m = (m_1, m_2) \in \mathbb{Z}^2, k = 1, 2, 3\}$$

is an orthonormal basis for the wavelet space $W_j^2$, given by $V_j^2 \oplus W_j^2 = V_{j-1}^2$. Moreover,

$$\{\psi_{j,m}^k \colon j \in \mathbb{Z}, m = (m_1, m_2) \in \mathbb{Z}^2, k = 1, 2, 3\}$$

is an orthonormal basis for $L^2(\mathbb{R}^2)$.

*Proof.* One can find the proof on [43], pp. 340-346. □

There exists more general non-separable two dimensional wavelets transforms using the continuous affine group to generalize the dilation operator $D_a$ to $D_M$ for two-dimensional invertible matrices $M$. The traditional theory of wavelets is based on the use of isotropic dilations and therefore is esentially a one-dimensional theory, so it is unable to give additional information about the geometry of the set of singularities of a function or distribution that are multivariate. The main problem is that the isotropic wavelet transform is simple but lacks of directional sensitivity and the ability to detect the multidimensional geometry of a function or distribution $f$.

One can formalize the notion of approximation quality using the concept of best $N$-term approximation. We will provide the general definition applied to dictionaries (collection of vectors on a Hilbert space $\{\varphi_i : i \in I\} \subseteq \mathcal{H}$ with $I$ finite or countable infinite).

**Definition 3.1** (Best N-term Approximation)**.** Let $D := \{\varphi_i \colon i \in I\} \subseteq \mathcal{H}$ be a dictionary. Consider a vector $x \in \mathcal{H}$ and an integer $N \in \mathbb{N}$. Then the *best N-term approximation of* $x$ with respect to $D$ is defined by the solution of the following minimization problem:

$$\min_{I_N, (c_i)_{i \in I_N}} ||x - \sum_{i \in I_N} c_i \varphi_i|| \text{ subject to } I : N \subseteq I, \#I_N \leq N$$

The best N-term approximation $f_N$ of $f \in L^2(\mathbb{R}^2)$ with respect to the dictionary formed by the wavelet basis can be understand as the obtained by approximating $f$ from its $N$ largest wavelet coefficients in magnitude. Let $\Lambda_N$ the index set corresponding to the $N$- largest wavelet coefficients $|\langle f, \psi_\lambda \rangle|$ associated with some wavelet basis $(\psi)_{\lambda \in \Lambda}$, the best $N$-term approximation will be

$$f_N = \sum_{\lambda \in \Lambda_N} \langle f, \psi_\lambda \rangle \psi_\lambda$$

To study the approximation of natural images by the wavelets, we first need to introduce a definition of what we will understand mathematically as a natural image, the so called *cartoon-like functions*.

**Definition 3.2** (Cartoon-like functions)**.** The class of *cartoon-like functions* $\mathcal{E}^2(\mathbb{R}^2)$ is defined as the set of functions $f : \mathbb{R}^2 \longrightarrow \mathbb{C}$ of the form $f = f_0 + \chi_B f_1$. Here, we assume that $B \subseteq [0,1]^2$ where $\partial B \in C^2$ and bounded curvature. Moreover, $f_i \in C^2(\mathbb{R}^2)$ with $||f_i||_{C^2} \leq 1$ and $\text{supp} f_i \subset [0,1]^2$ for $i = 0, 1$.

Now, let $f$ be a cartoon-like image containing a singularity along a smooth curve and $\{\psi_{j,m}\}$ be a standard wavelet basis of $L^2(\mathbb{R}^2)$. For $j$ sufficiently large, the only

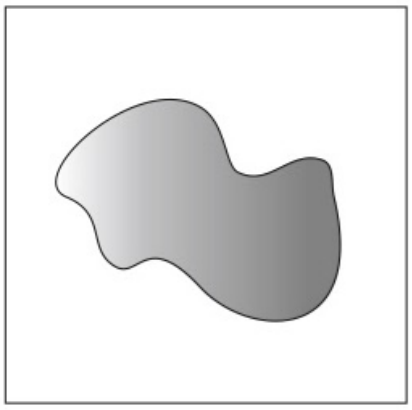


FIGURE 3.1: Example of a cartoon-like image. Figure taken from [44] pp. 9

significant wavelet coefficients $\langle f, \psi_{j,m} \rangle$ are the ones associated with the singularity. At each scale $2^{-j}$, each wavelet $\psi_{j,m}$ is supported inside a box of size $2^{-j} \times 2^{-j}$, there exist about $2^j$ elements of the wavelet basis overlapping the singularity curve. The associated wavelet coefficients are controlled by

$$|\langle f, \psi_{j,m} \rangle| \leq ||f||_\infty ||\psi_{j,m}||_{L^1(\mathbb{R}^2)} \lesssim 2^{-j}$$

It follows that the $N$-th largest wavelet coefficient in magnitude, denoted by $\langle f, \psi_{j,m} \rangle_{(N)}$, is bounded by $\mathrm{O}(N^{-1})$ (since $N \leq 2^j$). Thus, if $f$ is approximated by its best $N$-term approximation $f_N$, the $L^2$ error (called best $N$-term approximation error) obeys

$$\sigma_N(f, \{\psi_{j,m}\}_{j,m})^2 = ||f - f_N||^2_{L^2(\mathbb{R}^2)} \leq \sum_{\ell \geq N} |\langle f, \psi_{j,m} \rangle_{(l)}|^2 \lesssim N^{-1}$$

This estimate is actually tight, in the sense that there exist cartoon-like images for which the best $N$-term approximation error is

$$\sigma_N(f, \{\psi_{j,m}\}_{j,m}) \approx N^{-\frac{1}{2}}$$

the proof of this result can be founded in [43].

Even this looks like a nice result, it is far from optimal.

**Theorem 3.2.** Let $\{\psi_\lambda\}_{\lambda \in \Lambda} \subseteq L^2(\mathbb{R}^2)$ be a frame for $L^2(\mathbb{R}^2)$. Then the optimal best $N$-term approximation error for any $f \in \mathcal{E}^2(\mathbb{R}^2)$ is

$$\sigma_N(f, \{\psi_\lambda\}_{\lambda \in \Lambda}) = O(N^{-1})$$

*Proof.* In Section 3.2 we will define the concept of frame. This result was proved by Donoho in 2001 on [46], so one can refer to his proof. □

As we mentioned before, the problem with wavelets that does not make them to approach efficiently multivariate data is related to its isotropic scaling characteristic that makes them not sensible to directions. The question that can arise is, "Why should we care about anisotropic features related to multidimensional singularities?"; all the multivariate data are typically dominated by anisotropic features such as singularities on lower dimensional embedded manifolds; for example by edges in natural images or shock fronts in the solutions of transport equations.

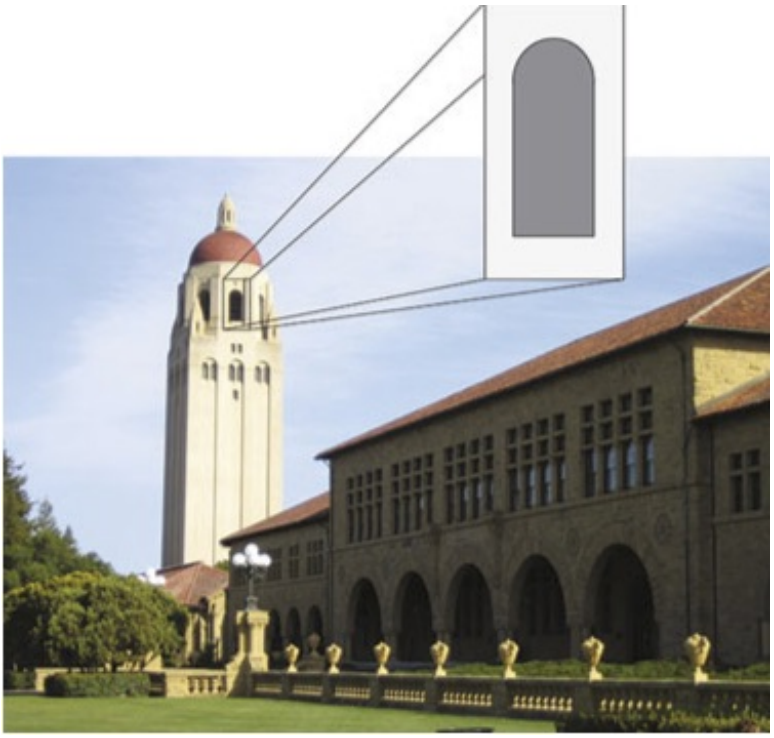


FIGURE 3.2: Natural images governed by anisotropic structures. Figure taken from [44] pp. 8

The bound result of theorem 3.2 works as a benchmark for optimally sparse approximation of two-dimensional data in form of cartoon-like functions. Moreover, to proof theorem 3.2 Donoho used adapted triangulations, which suggests that analyzing elements with elongated and orientable supports are required to get optimally sparse approximations of piecewise smooth two-dimensional functions. This observation leaded to two different approaches for solving this problem, the curvelets (proposed by E. Candès and D. Donoho in 1999 [47]), and the shearlets (proposed by Kanghui Guro, Gitta Kutyniok and Demetrio Labate in 2005 [48]), both are able to achieve the same optimal approximation rate; the one used in this thesis to sparsely represent EPIs is the latter due the possibility to develope a faithful implementation.

## 3.1 Shearlet Systems and Transform

We just discussed the limitations of wavelet systems in higher dimensions, we will then the concept of shearlet systems as a framework to solve these limitations. We also mentioned that in order to achieve optimally sparse approximations of signals with anisotropic singularities such as cartoon-like images, the analyzing elements must be made by waveforms ranging over several scales, orientations, and locations with the ability to become very elongated. One need then the combination of an appropriate scaling operator to generate elements at different scales, an orthogonal operator to change their orientations, and a translation operator to displace the elements over the two-dimensional plane.

By tradition and effectivenes one can use the family of dilation operators $D_{A_a}$, $a > 0$ based on parabolic scaling matrices $A_a$ of the form

$$A_a := \begin{pmatrix} a & 0 \\ 0 & a^{1/2} \end{pmatrix} \tag{3.1}$$

This is the first approach to a scaling operator by the long history of parabolic scaling in harmonic analysis literature [49]; the so called *Classical Shearlets* use this approach, one can generalize the scaling using matrices of the form

$$A_a := \begin{pmatrix} a & 0 \\ 0 & a^{\alpha/2} \end{pmatrix} \tag{3.2}$$

with $\alpha \in (0, 2)$ that controls the "degree of anisotropy" and the generated system is known as *Alpha Shearlets*, we will discuss this in detail on Section 3.3. Parabolic scaling is also known to be required in order to obtain optimally sparse approximations of cartoon-like images, since it is the best adapted to $C^2$-regularity of the curves of discontinuity, i.e. is efficient to approximate smooth curves, moreover choosing $a = 2$ gives the best performance.

Next, we need an orthogonal transformation to change to change the orientation of the waveforms. One does not use rotations since it destroys the structure of the integer lattice $\mathbb{Z}^2$ whenever the rotation angle is different from $0, \pm\frac{\pi}{2}, \pm\frac{3\pi}{2}$, which will represent an issue in the discrete setting. One chooses the shearing operator $D_s$, $s \in \mathbb{R}$, where the *shearing matrix* $S_s$ is given by

$$S_s = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \tag{3.3}$$

with this two elements we are ready to define the Continuous Shearlet Transform.

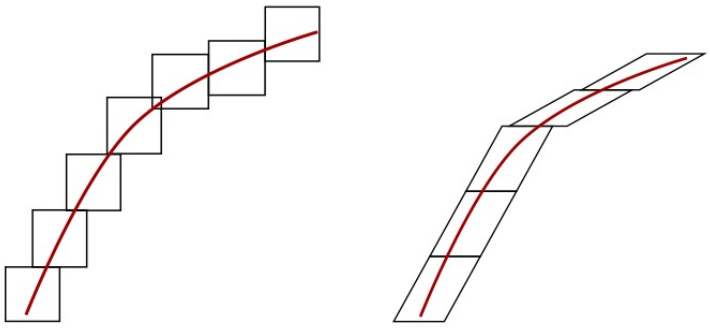


FIGURE 3.3: Optimal covering of anisotropic scaled and sheared atoms

**Definition 3.3** (Continuous Shearlet Transform)**.** Let $\psi \in L^2(\mathbb{R}^2)$, $A_a$ and $S_s$ the parabolic scaling matrix and shearing matrix defined in 3.1 and 3.3 respectively, then the continuous shearlet system $SH(\psi)$ associated with $\psi$ is defined by

$$\mathcal{SH}(\psi) := \{\psi_{a,s,t} = a^{1/2}\psi(S_s A_a x - t) : a \in \mathbb{R}^+, s \in \mathbb{R}, t \in \mathbb{R}^2\} \tag{3.4}$$

In analogy with the discretization of wavelets, one can discretize faithfully the shearlet system which permits a straight forward implementation. We will use $a = 2$ for scaling parameter since it was proven to be the best choice.

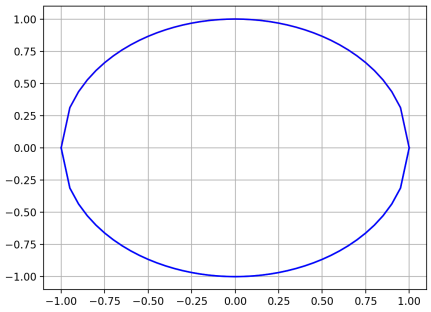



FIGURE 3.4: Circle before parabolic scaling

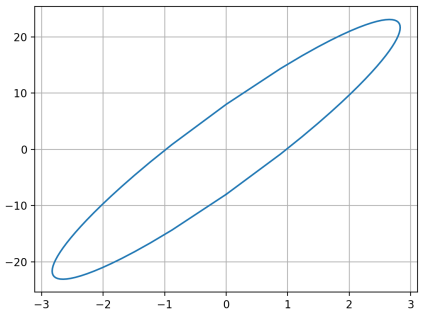



FIGURE 3.5: Circle after parabolic scaling $a = 4$

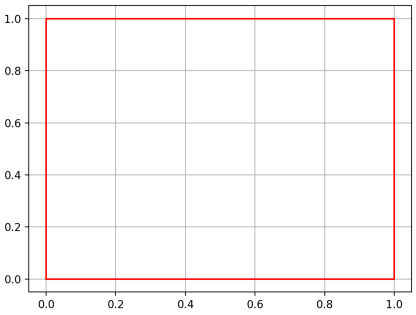



FIGURE 3.6: Square before shearing

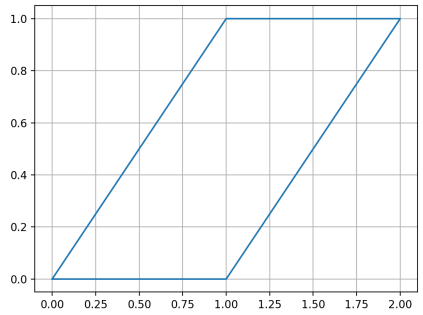



FIGURE 3.7: Sheared square by a factor of $k = 1$

**Definition 3.4** (Discrete Shearlet Transform)**.** Let $\psi \in L^2(\mathbb{R}^2)$, $j \in \mathbb{Z}$, lets define the *discrete parabolic scaling matrix* as follows

$$A_j := A_2^j = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix} \tag{3.5}$$

and the *discrete shearing matrix* for $k \in \mathbb{Z}$

$$S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix} \tag{3.6}$$

Given $\psi \in L^2(\mathbb{R}^2)$, the discrete shearlet system associated with $\psi$ is defined as

$$\mathcal{DSH}(\psi) := \{\psi_{j,k,m}(x) = 2^{3j/4}\psi(S_k A_j x - m) : j \in \mathbb{Z}, k \in \mathbb{Z}, m \in \mathbb{Z}^2\} \tag{3.7}$$

It has been proven that Shearlets present a lot of features and breakthrough results; for instance they can perform common tasks of signal processing as inpainting or denoising with great results in comparison with other methods (e.g. wavelets); but we defined the Shearlet System with motivation on the optimal best $N$-term approximation error found by Donoho (see Theorem 3.2), and prove that this bound is reached one first need to give some definitons.

**Definition 3.5** (Classical shearlets)**.** Let $\psi \in L^2(\mathbb{R}^2)$ be defined by

$$\hat{\psi}(\xi_1, \xi_2) = \hat{\psi}_1(\xi_1)\hat{\psi}_2\left(\frac{\xi_2}{\xi_1}\right)$$

where $\psi_1, \psi_2 \in L^2(\mathbb{R})$ satisfy the following properties:

- $\sum_{j\in\mathbb{Z}} |\hat{\psi}_1(2^{-j}\xi)|^2 = 1$ for a.e. $\xi \in \mathbb{R}$ (*wavelet like*).
- $\text{supp}(\hat{\psi}_1) \subseteq \left[\frac{1}{2}, -\frac{1}{16}\right] \cup \left[\frac{1}{16}, \frac{1}{2}\right]$
- $\hat{\psi}_1 \in C^\infty(\mathbb{R})$.
- $\sum_{k=-1,0,1} |\hat{\psi}_2(\xi + k)|^2 = 1$ for a.e. $\xi \in [-1, 1]$ ("bump-like").
- $\text{supp}(\hat{\psi}_2) \subseteq [-1, 1]$.
- $\hat{\psi}_2 \in C^\infty(\mathbb{R})$.

Then, we call $\psi$ a *classical shearlet*.

One can observe in Figure 3.8 that the tiling of the Fourier domain of the classical shearlets is not uniform at all, it is very biased towards the $\xi_2$-axis, that will lead to some issues if one wants to analyze singularities aligned with the $x_1$-axis. For directional systems as the shearlet system one would like to have a uniform tiling of the Fourier space; to achieve this one can split the space in "cones" and associate different shearlet system for each cone.

**Definition 3.6** (Cone-adapted shearlet system)**.** Let $\phi, \psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ and $c = (c_1, c_2) \in (\mathbb{R}^+)^2$. Lets split the Fourier space in cones $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ and a central low-frequency

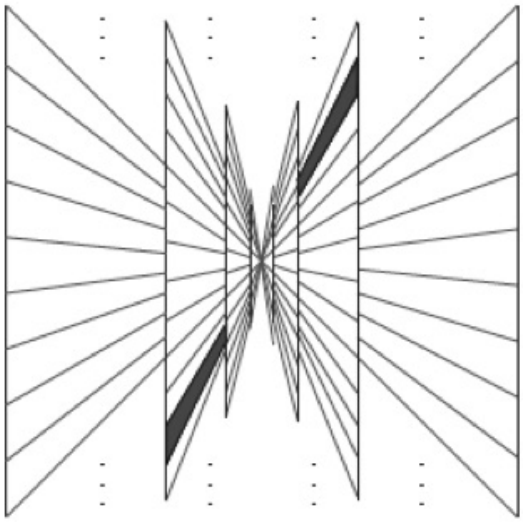


FIGURE 3.8: Tiling of the Fourier domain for the classical shearlets.
Figure taken from [50], pp. 82

square $\mathcal{R}$, see Figure 3.9, where

$$\begin{aligned}\mathcal{C}_1 \cup \mathcal{C}_3 := \mathcal{C}_h &= \{(\xi_2, \xi_1) \in \mathbb{R}^2 || \xi_2/\xi_1| \leq 1, |\xi_1| > 1\} \\ \mathcal{C}_2 \cup \mathcal{C}_4 := \mathcal{C}_v &= \{(\xi_1, \xi_2) \in \mathbb{R}^2 || \xi_2/\xi_1| > 1, |\xi_2| > 1\}, \\ \mathcal{R} &:= \{(\xi_1, \xi_2) \in \mathbb{R}^2 || \xi_1|, |\xi_2| \leq 1\}\end{aligned}$$

Then the *cone-adapted shearlet system* associated with $\phi, \psi, \tilde{\psi}$ and $c$ is defined by

$$\mathcal{SH}(\phi, \psi, \tilde{\psi}, c) := \mathcal{P}_{\mathcal{R}}\Phi(\phi, c_1) \cup \mathcal{P}_{\mathcal{C}_1}\Psi(\psi, c) \cup \mathcal{P}_{\mathcal{C}_2}\tilde{\Psi}(\tilde{\psi}, c)$$

where

$$\begin{aligned}\Phi(\phi, c_1) &:= \{\phi(x - c_1 m) \colon m \in \mathbb{Z}\}, \\ \Psi(\psi, c) &:= \{2^{3j/4}\psi(S_k A_j x - M_c m) \colon j \geq 0, |k| \leq \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\}, \\ \tilde{\Psi}(\tilde{\psi}, c) &:= \{2^{3j/4}\tilde{\psi}(\tilde{S}_k \tilde{A}_j x - \tilde{M}_c m) \colon j \geq 0, |k| \leq \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\},\end{aligned}$$

with

$$M_c = \begin{pmatrix} c_1 & 0 \\ 0 & c_2 \end{pmatrix}, \tilde{M}_c = \begin{pmatrix} c_2 & 0 \\ 0 & c_1 \end{pmatrix}, \tilde{S}_k = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix}, \tilde{A}_j = \begin{pmatrix} 2^{j/2} & 0 \\ 0 & 2^j \end{pmatrix}$$

and $P_{\mathcal{R}}$, $\mathcal{P}_{\mathcal{C}_1}$ and $\mathcal{P}_{\mathcal{C}_2}$ are the projections in the Fourier domain.

In the case of the wavelet transform the discretization and implementation of the algorithm that performs it is straight-forward since dilation can be performed by sub-sampling and one can use fourier transform properties to translate convolution into multiplication and that can be implemented optimally with the fft-algorithm. One can also take in account that the convolution operation with a function of compact support can be thought as a filtering operation in signal processing, so a wavelet system and a multeresolution analysis is interpreted in the typical implemenation as a filter bank, the last using a high pass and a low pass filter that characterize the interaction of the wavelet and scaling function, the former obeying the admissibility condition.

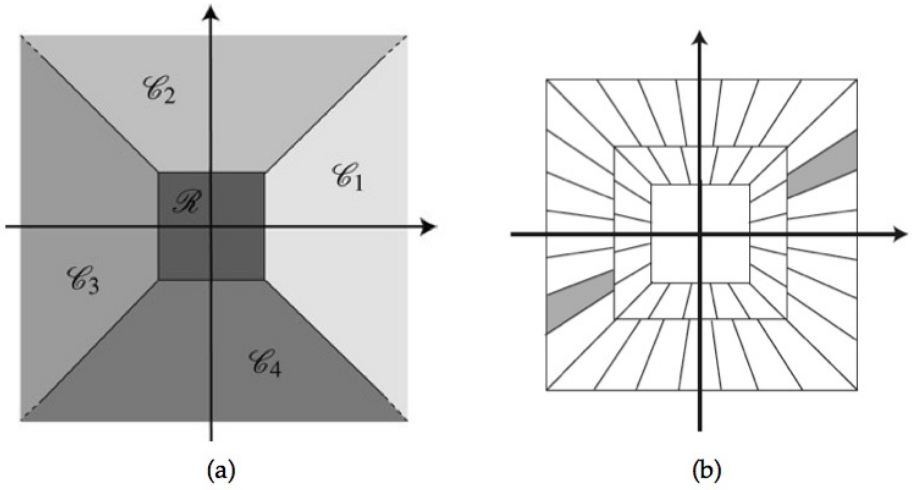



FIGURE 3.9: (a) Tiling of the Fourier domain into cones. (b) Frequency tiling generated by cone-adapted shearlets. Figure taken from [50] pp. 83

For the same reasons the Shearlet Transform of a function $f \in L^2(\mathbb{R}^2)$, defined by its coefficients $\langle f, \psi_{j,k,m} \rangle$ has a faithfull implementation using filtering and subsampling operations just taking care of the invariance of the $\mathbb{Z}^2$ grid under the discretization of the shearing operator (Wang-Q. Lim proposed a solution for this on [52]); the filters need to characterize the functions $\phi$ and $\psi$ on the cone-adapated shearlet system (Definition 3.6). The simplest choice is to take $\psi$ to be the tensor product of a wavelet function $\psi_1$ and a scaling function $\phi_1$ related to a multiresolution analysis, this approach is known as the separable shearlet transforms with generating functions:

$$\phi(x_1, x_2) = \phi_1(x_1)\phi_1(x_2)$$
$$\psi(x_1, x_2) = \psi_1(x_1)\phi_1(x_2)$$

Even this separable approach forms a frame a frame for $L^2(\mathbb{R}^2)$ (check Section 3.2 for a detail explanation) and simplifies the implementation, it is not a good choice for directional representations; the separability causes a significant overlap between $\text{supp}(\hat{\psi}_{j,k,m})$ and $\text{supp}(\hat{\psi}_{j,k+1,m})$, see Figure 3.10. Also in Figure 3.10 one can see that wedge shaped support is well adapted for covering the frequency domain by the application of the shear and scale operator while improving directional selectivity to achieve this Wang-Q. Lim proposed in 2013 a non-separable shearlet generator $\psi^{\text{non}}$ given by the relation

$$\hat{\psi}^{\text{non}}(\xi) = P\left(\frac{\xi_1}{2}, \xi_2\right)\hat{\psi}(\xi),$$

where $\psi$ is the already mentioned separable shearlet generator and $P$ is a 2D directional fan filter (see [52] for a more detailed explanation of this filter). The wedge form that non-separable shearlet generators give to the shearlet system the ability to cover the Fourier domain optimally. We will use this approach in this thesis; the implementation of the non-separable shearlet transform is widely explained in [51] where the most known implemenation is based on, Shearlab3D in matlab (one can download it in `https://shearlab.org`); by the improvement of performance we will use the Julia Programming Language (`https://julialang.org/`) implementation of this library that can be downloaded in `https://github.com/arsenal9971/Shearlab.jl`

or installed from the Julia REPL using Pkg.add("Shearlab").

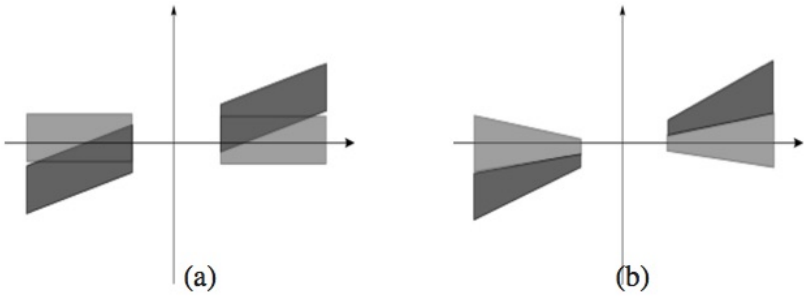



FIGURE 3.10: Frequency covering by shearlets $\psi_{j,0,m}$ and $\psi_{j,1,m}$ (a) with separable generator (b) with non-separable generator. Figure taken from [52] pp. 12

We finally can give in this section the result that we were looking for to justify formally the superiority of the Shearlet System over the Wavelet System on representating optimally the *cartoon-like functions*, we will again make use of the term frame, despite it has not been introduced yet, but until the next section.

**Theorem 3.3.** Let $\psi \in L^2(\mathbb{R}^2)$ be compactly supported.

Assume that for $\alpha > 5$, $\gamma \geq 4$, $q > q' > 0$ and $q > r > 0$, it holds that

$$|\hat{\psi}(\xi_1, \xi_2)| \leq C \min\{1, |q\xi_1|^\alpha\} \min\{1, |q'\xi_1|^{-\gamma}\} \min\{1, |r\xi_2|^{-\gamma}\} \tag{3.8}$$

for some constant $C > 0$ and that for $h \in L^1(\mathbb{R}^1)$

$$|\frac{\partial}{\partial \xi_2}\hat{\psi}(\xi)| \leq |h(\xi_1)| \left(1 + |\frac{\xi_2}{\xi_1}|\right)^{-\gamma}$$

is satisfied. Assume that the same conditions are satisfied for $\tilde{\psi}$. Then, if the *cone-adapated shearlet system* on definition 3.6 $\mathcal{SH}(\phi, \psi, \tilde{\psi}, (1,1))$ forms a frame for $L^2(\mathbb{R}^2)$, then there exists a constant $C' > 0$ such that for all $f \in \mathcal{E}^2(\mathbb{R}^2)$, we have

$$\sigma_N(f, \mathcal{SH}(\phi, \psi, \tilde{\psi}, (1,1)) \leq C' N^{-1} (\log N)^{3/2} \text{as } N \longrightarrow \infty \tag{3.9}$$

*Proof.* The proof of this theorem is quite technical and it has not a great impact on the results of our thesis, so one will let the reader to consult [48] for a detailed proof. □

As $N$ goes bigger the logarithm in the estimate 3.9 can be taken as a constant, so Theorem 3.3 shows that the Cone-Adapted Shearlet System atains the theoretical optimal best $N$-term approximation error for the cartoon like functions given in Theorem 3.2. Cartoon-like functions represent accurately natural images, so this system is proven to be a great option to inpaint EPIs in this thesis. In the next sections we will show other features of the Shearlets as its character of frames for $L^2(\mathbb{R}^2)$ as well as other forms of them using general scaling matrix, called alpha shearlets and we will show why a particular case of alpha shearlets is the best to inpaint sparse-sampled EPIs.

## 3.2 Shearlets as Frames

Lets now take a bigger picture on representation systems. An orthonormal basis for Hilbert space $\mathcal{H}$ is a sequence $(\phi_i)_{i\in I} \subset \mathcal{H}$ such that for each vector $x \in \mathcal{H}$

$$x = \sum_{i\in I} \langle x, \phi_i \rangle \phi_i \tag{3.10}$$

so one can represent each vector in terms of the collection and one can decompose each element $x \in \mathcal{H}$ as

$$x \mapsto (\langle x, \phi_i \rangle)_{i\in I},\ \mathcal{H} \longrightarrow \ell_2(I)$$

even this two features (decomposition and recovery) are very simply expressed for orthonormal bases, one woul like to extend the theory for different reasons. For example, it is not possible to recover $x$ if one loses some of the coefficients $\langle x, \phi_i \rangle$, so the induced decomposition is not robust. In the last section we mentioned the imporance of sparse representation of signals in different applications, but an orthonormal basis forces the representation coefficients to be $\langle x, \phi_i \rangle$, the sequence of coefficients will not have a rapid decay. One will call this two mentioned characteristics as *Robust Decomposition*, and *Sparse representation*.

A natural generalization of the concept of orthonormal bases is the concept of frames, to define them one weakens the Parseval equation 3.10.

**Definition 3.7** (Frames)**.** (1) A sequence $(\phi_i)_{i\in I}$ in a Hilbert space $\mathcal{H}$ is called a **frame** for $\mathcal{H}$, if there exist constans $0 < A \leq B < \infty$ such that

$$A||x||^2 \leq \sum_{i\in I} |\langle x, \phi_i \rangle|^2 \leq B||x||^2,\ \forall x \in \mathcal{H} \tag{3.11}$$

$A$ and $B$ are called lower and upper frame bound.

(2) If $A$ and $B$ can be chosen to be equal, we call it ($A$-) tight frame. If $A = B = 1$ is possible, $(\phi_i)_{i\in I}$ forms a Parseval frame.

(3) If only the upper bound in 3.11 holds, we call $(\phi_i)_{i\in I}$ a Bessel sequence.

With this definition an orthonormal basis will be a Parseval frame. Since $A > 0$ frames also span the whole space $\mathcal{H}$, frames allows one to obtain both characteristics *robust decomposition* and *sparse representation*; for a detailed explanation of this we highly recommend the Chapter 5 of [43].

In the last section we introduced different representation systems for signals in $L^2(\mathbb{R}^2)$, as Gabor systems that emerge motivated by the short time fourier transform, wavelet and shearlet systems. This systems will form a frame under certain conditions on the generating functions and parameters. For example in the case of wavelet systems we have the next theorem that gives a necessary condition to form frames.

**Theorem 3.4** (Necessary condition for wavelet frames)**.** Let $a > 1$, $b > 0$ and $\psi \in L^2(\mathbb{R})$ a wavelet function such that the related system $W(\psi, a, b)$ forms a frame for $L^2(\mathbb{R})$ with frame bounds $A$ and $B$. Then, we have

$$A \leq \frac{1}{b} \sum_{j\in\mathbb{Z}} |\hat{\psi}(a^j \xi)|^2 \leq B,\ \forall \xi \in \mathbb{R}$$

*Proof.* One can find the proof in Theorem 3.3 of [53] □

Sufficient conditions of wavelet frames are more technical and one can find them in Theorem 3.15 of [50]. In our case we would like to know under what conditions *Classical Shearlets* and *Cone-adapted shearlets* separable and nonseparable form frames, and for that we have the next results.

**Theorem 3.5.** Let $\psi$ be a classical shearlet, i.e. obeys the conditions 3.5. Then the associated wavelet system $\mathcal{SH}(\psi)$ forms a Parseval frame for $L^2(\mathbb{R}^2)$.

*Proof.* By the above properties of $\psi_1$ and $\psi_2$, we have

$$\begin{aligned}\sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}}|\hat{\psi}(S_{-k}^T A_{-j}\xi)|^2 &= \sum_{j\in\mathbb{Z}}|\hat{\psi}_1(2^{-j}\xi_1)|^2\sum_{k\in\mathbb{Z}}|\hat{\psi}_2(2^{j/2}\xi_2/\xi_1-k)|^2\\ &= \sum_{j\in\mathbb{Z}}|\hat{\psi}_1(2^{-j}\xi_1)|^2 = 1\end{aligned}$$

this holds for almost every $\xi\in\mathbb{R}^2$, using on this equation Plancherel's and Parseval's identity is enough to finish the proof. □

Using a similar proof of the last theorem one can prove the next theorem.

**Theorem 3.6.** Let $\psi\in L^2(\mathbb{R}^2)$ be a classical shearlet. Then

$$\Psi(\psi,(1,1)) := \{2^{3j/4}\psi(S_kA_jx-m)\colon\ j\geq 0, |k|\leq\lceil 2^{j/2}\rceil, m\in\mathbb{Z}^2\}$$

forms a Parseval frame for

$$\{f\in L^2(\mathbb{R})\colon \operatorname{supp}\hat{f}\subset\{\xi\in\mathbb{R}^2\colon\ |\xi_1|\geq 1, |\xi_2/\xi_1|\leq 1\}\}$$

Finally one can state the result for the cone-adapted shearlet system.

**Theorem 3.7.** For $\alpha>\gamma>3$, $q>q'>0$ and $q>r>0$, let

$$|\hat{\psi}(\xi_1,\xi_2)|\leq C\min\{1,|q\xi_1|^\alpha\}\min\{1,|q'\xi_1|^{-\gamma}\}\min\{1,|r\xi_2|^{-\gamma}\} \tag{3.12}$$

for some constant $C>0$, and that

$$\sum_{j,k\in\mathbb{Z}}|\hat{\psi}(S_{-k}^TA_{-j}\xi)|^2\geq C'>0$$

for almost every $\xi\in\mathbb{R}^2$. For $\tilde{\psi}$, we assume similar conditions. Then, there exists some $c_0$ such that $\mathcal{SH}(\phi,\psi,\tilde{\psi},c)$, with suitable $\phi$, forms a frame for $L^2(\mathbb{R}^2)$ for all $c_1,c_2<c_0$ and for the frame bounds, we have

$$C_1(\alpha,\gamma,q,q',r,c_1,c_2)\leq A\leq B\leq C_2(\alpha,\gamma,q,q',r,c_1,c_2)$$

*Proof.* We refer to [48] for the proof of this theorem. □

The choice of functions $\psi$ and $\phi$ presented in the last section in both separable and non-separable fashion obey the inequality 3.12 (see [52]); therefore the related shearlet system will form a frame. This machinery shows the strong properties of the shearlet systems being frames, presenting both sparse representation and robust decomposition and in particular shows that the shearlets will span $L^2(\mathbb{R}^2)$. In the next section we will extend the notion of scaling that is suitable for levels of anisotropy.

## 3.3 Universal Shearlets and Alpha Shearlets

The classical and the cone-adapted shearlets that we mentioned so far make use of a parabolic scaling matrix to perform the scaling operation, this matrix has the form

$$A_j = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix} \tag{3.13}$$

the form of the matrix has been motivated by the aim to construct best approximation of functions with singularities over parabolic curves, but one would like to have more flexibility on curves that dominate the images; in the case of the Epipolar Images as one can see in Section 2.4 one would like to have a system that is good to approximate straight line singularities.

The natural generalization of the parabolic scaling was motivated by the inpainting problem (see [2]) is reached by the introduction the so called *scaling sequences* $(\alpha_j)_j \subseteq (-\infty, 2)$ with associated scaling matrices

$$A_{j,\alpha_j,(h)} = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{\alpha_j j/2} \end{pmatrix} \tag{3.14}$$

which offers a lot of flexibility in scaling and let us choose the level of anisotropy for each scale $j$ independently; using definition of scaling matrix we will define the *Universal Shearlet System* introduced by Gitta Kutyniok and Martin Genzel in [2]; it is very convinient to generalize the notion of scaling also because it permits us to have a uniform treatment of different band-limited systems like classical cone-adapted shearlets, ridgelets and band-limited wavelets.

One main objective of the introduction of the *Universal Shearlet System* is to construct due to its flexibility a compactly supported directional system which forms a Parseval frame and gives an optimal sparsifying approximation of cartoon-like functions. The classical shearlet theory showed that certain class of band-limited shearlets constitute a Parseval frame and provide an optimal sparse approximation (see [54]); but, when trying to force the compact support of those even under certain conditions optimal approximation rates can be achieved, they weill not have the Parseval property.

Lets proceed with the construction of the *Universal Shearlets*. First let us recall the definition of the *Schwartz functions space*

$$\mathbb{S}(\mathbb{R}^d) := \{\phi \in C^\infty(\mathbb{R}^d) | \forall K, N \in \mathbb{N}_0 : \sup_{x \in \mathbb{R}^d} (1 + |x|^2)^{-N/2} \sum_{|\alpha| \leq K} |D^\alpha \phi(x)| < \infty\}$$

we will rather use the compact notation $\langle |x| \rangle := (1+|x|^2)^{-N/2}$, the fourier transform will be an operator of $\mathbb{S}(\mathbb{R}^d)$.

Lets define a scaling function as $\phi \in \mathbb{S}(\mathbb{R})$ satisfying $0 \leq \hat{\phi} \leq 1$, $\hat{\phi}(u) = 1$ for $u \in [-1/16, 1/16]$, and $\text{supp}(\hat{\phi}) \subset [-1/8, 1/8]$. A function with these properties is usually named *Meyer scaling function*. Now, lets define the *corona scaling function* for $j \in \mathbb{N}_0$ by ($\xi = (\xi_1, \xi_2) \in \mathbb{R}^2$)

$$\hat{\Phi}(\xi) := \hat{\phi}(\xi_1)\hat{\phi}(\xi_2),$$
$$W(\xi) := \sqrt{\hat{\Phi}^2(2^{-2}\xi) - \hat{\Phi}^2(\xi)},$$
$$W_j(\xi) := W(2^{-2j}\xi).$$

The functions $W_j$ are compactly supported in corona-shaped scaling levels, i.e.

$$\text{supp}W_j \subset \mathcal{K}_j := [-2^{2j-1}, 2^{2j-1}]^2 \setminus (-2^{2j-4}, 2^{2j-4})^2 \tag{3.15}$$

and as in the case of classical cone-adapted shearlets one can decompose the Fourier domain by the sequence of scaling functions:

$$\hat{\Phi}^2(\xi) + \sum_{j\geq 0} W_j^2(\xi) = 1,\ \xi \in \mathbb{R}^2$$

as in the definiton of the classical shearlets the function $W$ can be viewed as a wavelet, in the same manner one considers a *bump-like* function $v \in C^\infty(\mathbb{R})$ which satisfies supp$v \subset [-1, 1]$ and

$$|v(u-1)|^2 + |v(u)|^2 + |v(u+1)|^2 = 1 \quad \text{for} \quad u \in [-1, 1] \quad \text{,and} \tag{3.16}$$
$$v(0) = 1 \quad \text{and} \quad v^{(n)}(0) = 0 \quad \text{for} \quad n \geq 1 \tag{3.17}$$

For an explicit construction of $v$ we refer to [54], again to avoid directional bias along $\xi_2$-axis at the Fourier space we will use the cone-adapted approach (see definition 3.6), with

$$\mathcal{C}_{(h)} := \{(\xi_1, \xi_2) \in \mathbb{R}^2 | |\xi_2/\xi_1| \leq 1\}$$
$$\mathcal{C}_{(v)} := \{(\xi_1, \xi_2) \in \mathbb{R}^2 | |\xi_2/\xi_1| > 1\}$$

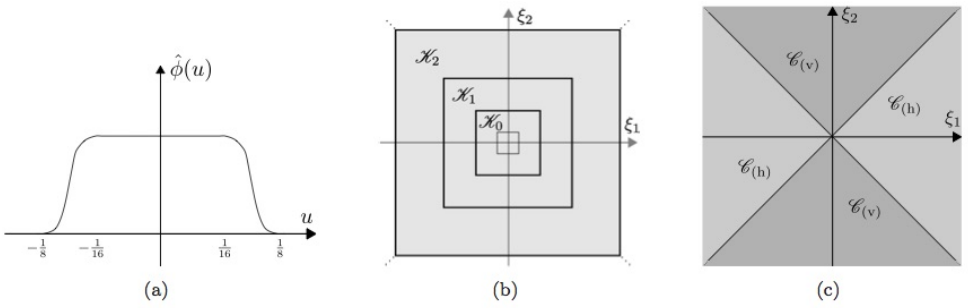



FIGURE 3.11: (a) Fourier transform of a Meyer scaling function. (b) Decomposition of the frequency plane by corona functions $W_j$ and $\Phi$. (c) Symmetric frequency decompositionby cones. Figure taken from [2] pp. 11

Lets introduce adapted versions of the usual shearing and scaling matrices,

$$A_{\alpha,(h)} := \begin{pmatrix} 2 & 0 \\ 0 & 2^{\alpha/2} \end{pmatrix} \quad , \quad S_{(h)} := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$A_{\alpha,(v)} := \begin{pmatrix} 2^{\alpha/2} & 0 \\ 0 & 2 \end{pmatrix} \quad , \quad S_{(v)} := \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

where $\alpha \in (-\infty, 2)$ is the *scaling parameter*, $A_{j,\alpha,(\iota)} := A^j_{\alpha,(\iota)}$ and $S_{k,(\iota)} := S^k_{(\iota)}$ for $\iota \in \{h, v\}$. The adapted *cone functions* are given by

$$V_{(h)}(\xi) := v(\xi_2/\xi_1), \quad V_{(v)}(\xi) := v(\xi_1/\xi_2), \quad \xi \in \mathbb{R}^2.$$

Lets define now the ingredients of a universal shearlet system.

**Definition 3.8.** Let $\Phi$, $W$, $V_{(h)}$, $V_{(v)} \in L^2(\mathbb{R}^2)$ be defined as before.

1. **Coarse scaling functions**: For $k \in \mathbb{Z}^2$, we set

$$\psi_{-1,k} := \Phi(x - k), \quad x \in \mathbb{R}^2.$$

2. **Interior shearlets**: Let $\alpha \in (-\infty, 2)$, $j \in \mathbb{N}_0$, $k \in \mathbb{Z}$ with $|k| < 2^{(2-\alpha)j/2}$, $m \in \mathbb{Z}^2$ and $\iota \in \{h, v\}$. The shearlets will be given by

$$\hat{\psi}^{\alpha,(\iota)}_{j,k,m}(\xi) := 2^{-(\alpha+2)j/4} W(2^{-j}\xi) V_{(\iota)}(\xi^\top A_{-j,\alpha,(\iota)} S_{-k,(\iota)}) e^{-2\pi i \xi^\top A_{-j,(\iota)} S_{-k,(\iota)}}, \quad \xi \in \mathbb{R}^2 \tag{3.18}$$

3. **Boundary shearlets**: For $\alpha \in (-\infty, 2)$, $j \geq 1$, $k = \pm\lceil 2^{(2-\alpha)j/2} \rceil$ and $k \in \mathbb{Z}^2$, we define

$$\hat{\psi}^{\alpha}_{j,k,m} := \begin{cases} 2^{-(\alpha+2)j/4 - 1/4} W(2^{-j}\xi) V_{(h)}(\xi^\top A_{-j,\alpha,(h)} S_{-k,(\iota)}) e^{-\pi i \xi^\top A_{-j,\alpha,(h)} S_{-k,(h)} m}, & \xi \in \mathcal{C}_{(h)}, \\ 2^{-(\alpha+2)j/4 - 1/4} W(2^{j}\xi) V_{(v)}(\xi^\top A_{-j,\alpha,(v)} S_{-k,(v)}) e^{-\pi i \xi^\top A_{-j,\alpha,(v)} S_{-k,(h)} m}, & \xi \in \mathcal{C}_{(v)} \end{cases} \tag{3.19}$$

and in the case $j = 0$, $k = \pm 1$, we define

$$\hat{\psi}^{\alpha}_{0,k,m} := \begin{cases} W(\xi) V_{(h)}(\xi^\top S_{-k,(h)}) e^{-2\pi i \xi^\top m}, & \xi \in \mathcal{C}_{(h)}, \\ W(\xi) V_{(v)}(\xi^\top S_{-k,(v)}) e^{-2\pi i \xi^\top m}, & \xi \in \mathcal{C}_{(v)}. \end{cases}$$

One can compare this definition of shearlet system with the Definition 3.6 of classical cone-adapted shearlet system and see that in the case of $\alpha = 1$ both are the same. The Fourier transform $\hat{\psi}^{\alpha,(h)}_{j,k,m}$ has compact support in the trapezoidal region

$$\{\xi \in \mathbb{R}^2 | \xi_1 \in [-2^{j-1}, 2^{j-1}] \setminus (-2^{j-2}, 2^{j-2}), |\xi_2/\xi_1 - k 2^{-(2-\alpha)j/2}| \leq 2^{-(2-\alpha)j/2}\}. \tag{3.20}$$

Since the boundary shearlets were defined piecewise, smoothness (of the Fourier transforms) is not guaranteed for all $\alpha \in (-\infty, 2)$. The exponent $2^{(2-\alpha)j/2}$ needs to be necessarily integer-valued when analyzing the first partial derivatives of eq. 3.19. To satisfy this for every $j \in \mathbb{Z}$, we would have to restrict the set of admissible $\alpha$ to the set of integers; but this would not give us the arbitrary scaling behavior that we are looking for. To overcome this issue one can relax the condition of a globally fixed $\alpha$ and, introduce a *separate* scaling parameter $\alpha_j$ on each scale. The universal shearlet system will be associated with a whole sequence $(\alpha_j)_{j \in \mathbb{N}_0} \subset \mathbb{R}$. In this setting the set of admissible scaling parameters can be substantially enlarged since we have $(2 - \alpha_j)j/2 \in \mathbb{Z}$ whenever $\alpha_j$ is multiple of $2/j$.

**Definition 3.9.** A sequence $(\alpha_j)_{j\in\mathbb{N}_0} \subset \mathbb{R}$ is called a scaling sequence if

$$\alpha_j \in A_j := \left\{ \frac{2n}{j} | n \in \mathbb{Z}, n \leq j-1 \right\} = \left\{ \ldots, -\frac{4}{j}, -\frac{2}{j}, 0, \frac{2}{j}, \ldots, 1-\frac{2}{j} \right\}$$

**Definition 3.10.** Let $(\alpha_j)_{j\in\mathbb{N}_0}$ be a *scaling sequence*. Then we define the associated universal-scaling shearlet system, or shorter, universal shearlet system, by

$$\mathcal{SH}(\phi, v, (\alpha_j)_j) := \mathcal{SH}_{\text{Low}}(\phi) \cup \mathcal{SH}_{\text{Int}}(\phi, v, (\alpha_j)_j) \cup \mathcal{SH}_{\text{Bound}}(\phi, v, (\alpha_j)_j),$$

where

$$\begin{aligned}
\mathcal{SH}_{\text{Low}}(\phi) &:= \{\psi_{-1,m} | m \in \mathbb{Z}^2\} \\
\mathcal{SH}_{\text{Int}}(\phi, v, (\alpha_j)_j) &:= \{\psi^{\alpha_j,(\iota)}_{j,k,m} | j \geq 0, |m| < 2^{(2-\alpha_j)j/2}, m \in \mathbb{Z}^2, \iota \in \{h, v\}\}, \\
\mathcal{SH}_{\text{Bound}}(\phi, v, (\alpha_j)_j) &:= \{\psi^{\alpha_j}_{j,k,m} | j \geq 0, |k| = \pm 2^{(2-\alpha_j)j/2}, m \in \mathbb{Z}^2\}.
\end{aligned}$$

Since the elements of a scaling sequence $\alpha_j$ can be chosen independently for each scaling level and, in particular, the sequence $(\alpha_j)_j$ does not need to converge; this permits a faithful implementation of the universal shearlet transform; this is already implemented in the toolbox used in this thesis (Shearlab.jl) one determines the total number of shearings (on each scale) first given by $|k| \leq \lceil 2^{(2-\alpha_j)j/2} \rceil$, rather that selecting a fixed $\alpha_j$. We are ready to prove the main property of the universal shearlet system, i.e. its character of frame.

**Theorem 3.8** (Universal shearlet frames)**.** Let $(\alpha_j)_j$ be a scaling sequence and $\mathcal{SH}(\phi, v, (\alpha_j)_j)$ be an associated universal shearlet system. Then $\mathcal{SH}(\phi, v, (\alpha_j)_j)$ constitutes a Parseval frame for $L^2(\mathbb{R}^2)$ consisting of band-limited Schwartz functions. Moreover, the interior and boundary shearlets have infinitely many vanishing moments.

*Proof.* Lets proceed with the proof of the different properties required.

- *Band-limiting and vanishing moment property:* Observing that $(0,0) \notin \text{supp} W_k \subset \mathcal{K}_j$ which immediately follows from Definition 3.8.
- *Smoothness:* Due to the band-limiting of $\mathcal{SH}(\phi, v, (\alpha_j)_j)$, it is sufficient to show that the Fourier transforms of the elements are smooth. This will imply that $\mathcal{SH}(\phi, v, (\alpha_j)_j) \subset \mathbb{S}(\mathbb{R}^2)$.

  The smoothness of $\mathcal{SH}_{\text{Low}}(\phi, v, (\alpha_j)_j)$ and $\mathcal{SH}_{\text{Int}}(\phi, v, (\alpha_j)_j)$ are induced by their smooth defining functions $\phi$ and $v$. We still need to consider elements $\psi^{\alpha_j}_{j,k,m} \in \mathcal{SH}_{\text{Bound}}(\phi, v, (\alpha_j)_j)$. In the interior of $\mathcal{C}_{(h)}$ and $\mathcal{C}_{(v)}$, the smoothness is again obvious. Thus, we only need to analyze the boundary lines of the cones which are given by $\{|\xi_1| = |\xi_2|\}$.

  Lets use the shortcut $k_j := 2^{(2-\alpha_j)j/2}$ for the maximal shearing number (on scale $j$) and simplify the definition of $\hat{\psi}^{\alpha_j}_{j,k,m}$ (for $j \geq 1$):

$$\hat{\psi}^{\alpha_j}_{j,k,m}(\xi) = 2^{-\frac{(2+\alpha_j)j}{4}-\frac{1}{4}} \cdot \begin{cases} W(2^{-j}\xi)v(k_j(\xi_2/\xi_1-1)) & e^{-\pi i[2^{-j}\xi_1 m_1 + 2^{-\alpha_j j/2}(\xi_2-\xi_1)m_2]} \\ & , \quad \xi \in \mathcal{C}_{(h)}, \\ W(2^{-j}\xi)v(k_j(\xi_1/\xi_2-1)) & e^{-\pi i[2^{-j}\xi_1 m_1 + 2^{-\alpha_j j/2}(\xi_1-\xi_1)m_2]} \\ & , \quad \xi \in \mathcal{C}_{(v)}, \end{cases} \tag{3.21}$$

In the case of $\xi_1 = \pm\xi_2$, both terms coincide implying the continuity of $\hat{\psi}^{\alpha_j}_{j,k,m}$. By the same argument, the continuity of $\hat{\psi}^{\alpha_j}_{j,-k_j,m}$ is verified. Next we compute the partial derivatives of both terms in eq. 3.21 which are given by

$$\begin{aligned} &\frac{\partial}{\partial\xi_1}\Big|_{\xi_1=\xi_2}[W(2^{-j}\xi)v(k_j(\xi_2/\xi_1-1))e^{-\pi i[2^{-j}\xi_1 m_1 + 2^{-\alpha_j j/2}(\xi_2-\xi_1)m_2]}] \\ &\quad = 2^{-j}\frac{\partial W}{\partial\xi_1}(2^{-j}\xi_1, 2^{-j}\xi_1)v(0)e^{-2^{-j}\pi i\xi_1 m_1} - \frac{k_j}{\xi_1}W(2^{-j}\xi_1, 2^{-j}\xi_1)v'(0)e^{-2^{-j}\pi i\xi_1 m_1} \\ &\qquad - \pi i(2^{-j}m_1 - 2^{-\alpha_j j/2}m_2)W(2^{-j}\xi_1, 2^{-j}\xi_1)v(0) \\ &\qquad \cdot e^{-2^{-j}\pi i\xi_1 m_1} \end{aligned} \tag{3.22}$$

and

$$\begin{aligned} &\frac{\partial}{\partial\xi_1}\Big|_{\xi_1=\xi_2}[W(2^{-j}\xi)v(k_j(\xi_1/\xi_2-1))e^{-\pi i[2^{-j}\xi_1 m_1 + 2^{\alpha_j j/2}(\xi_2-\xi_1)m_2]}] \\ &= 2^{-j}\frac{\partial W}{\partial\xi_1}(2^{-j}\xi_1, 2^{-j}\xi_1)v(0)e^{-2^{-j}\pi i\xi_1 m_1} + \frac{k_j}{\xi_1}W(2^{-j}\xi_1, 2^{-j}\xi_1)v'(0)e^{-2^{-j}\pi i\xi_1 m_1} \\ &\quad - \pi i(2^{-j}m_1 - 2^{-\alpha_j j/2}m_2)W(2^{-j}\xi_1, 2^{-j}\xi_1)v(0)e^{-2^{-j}\pi i\xi_1 m_1} \end{aligned} \tag{3.23}$$

These two expressions coincide, since by definition $v'(0) = 0$. The same can be done in a similar way for the partial derivative with respect to $\xi_2$, and by obvious modifications, one verifies the smoothness for $\hat{\psi}^{\alpha_j}_{j,-k_j,m}$ as well as for the case $j = 0$. Finally, the differentiability of higher order is proven by induction and successive use of eq. 3.17.

- *Parseval frame property:* Let $f \in L^2(\mathbb{R}^2)$ be arbitrary. Lets consider the different parts of $\mathcal{SH}(\phi, v, (\alpha_j)_j)$ separately:

**Case 1** Boundary shearlets with $j \geq 1$: By Plancherel's Theorem, we get

$$\sum_{m\in\mathbb{Z}^2} |\langle f, \psi^{\alpha_j}_{j,k,m}\rangle|^2 = \sum_{k\in\mathbb{Z}^2} \langle \hat{f}, \hat{\psi}^{\alpha_j}_{j,k,m}\rangle|^2 = \sum_{k\in\mathbb{Z}^2} \Big| \int_{\mathbb{R}^2} 2^{-(2+\alpha_j)j/4-1/2}\hat{f}(\xi)W(2^{-j}\xi)$$
$$ep^{\pi i\xi^\top A_{-j,\alpha_j,(h)}S_{-k_j,(h)}m} \times [\chi_{\mathcal{C}_{(h)}}V_{(h)}(\xi^\top A_{-j,\alpha_j,(h)}S_{-k_j,(h)}) + \chi_{\mathcal{C}_{(v)}}(\xi)V_{(v)}(\xi^\top A_{-j,\alpha_j,(v)}S_{k_j,(v)})]d\xi\Big|^2 \tag{3.24}$$

In order to apply the Parseval's identity, we can make use of that

$$\eta^\top := 2^{-1}\xi^\top A_{-j,\alpha_j,(h)}S_{-k_j,(h)} \Leftrightarrow \xi = \xi(\eta) = (2^{j+1}\eta_1, 2^{j+1}\eta_1 + 2^{\alpha_j j/2+1}\eta_2).$$

Then the equation 3.24 will have the form

$$\begin{aligned} V_{(h)}\left(\xi^\top A_{j,\alpha_j,(h)} S_{-k_j,(h)}\right) &= v\begin{pmatrix}\eta_2\\ \eta_1\end{pmatrix}, \\ V_{(v)}\left(\xi^\top A_{-j,\alpha_j,(v)} S_{-k_j,\alpha_j,(v)}\right) &= v\left(2^{(2-\alpha)j/2}(\xi_1/\xi_2 - 1)\right) \\ &= v\left(-\frac{\eta_2}{\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2}\right), \\ W(2^{-j}\xi) &= W\left(2\eta_1, 2[\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2]\right). \end{aligned}$$

By equation 3.15, the mapping $(\eta_1, \eta_2) \mapsto W(2\eta_1, 2[\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2])$ is supported in the strip $\{|\eta_1| \le 1/4\}$. Since $\mathrm{supp} v \subset [-1,1]$, we have that the mappings

$$\begin{aligned} (\eta_1,\eta_2) &\mapsto U_{(h),j}(\eta) := W(2\eta_1, 2[\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2]) v\left(\frac{\eta_2}{\eta_1}\right) \\ (\eta_1,\eta_2) &\mapsto U_{(v),j}(\eta) := W(2\eta_1, 2[\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2]) v\left(-\frac{\eta_2}{\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2}\right) \end{aligned}$$

are supported in the square $Q^2 := [-1/2, 1/2]^2$. Here, we used

$$\begin{aligned} \left|-\frac{\eta_2}{\eta_1 + 2^{(\alpha_j-2)j/2}\eta_2}\right| \le 1 &\Longrightarrow \left|\frac{\eta_2}{\eta_1}\right| \le \left|1 + 2^{(\alpha_j-2)j/2}\frac{\eta_2}{\eta_1}\right| \le 1 + 2^{(\alpha_j-2)j/2}\left|\frac{\eta_2}{\eta_1}\right| \\ &\Longrightarrow \left|\frac{\eta_2}{\eta_1}\right| \le \frac{1}{1 - 2^{(\alpha_j-2)j/2}} \le 2 \end{aligned}$$

where in the last inequality we used $\alpha_j \le 1 - 2/j$. With this observation, we can continue in 3.24 by

$$\begin{aligned} &\sum_{k\in\mathbb{Z}^2} |\langle f, \psi^{\alpha_j}_{j,k_j,m}\rangle|^2 \\ &= \sum_{k\in\mathbb{Z}^2} \Big| \int_{Q^2} 2^{(2+\alpha_j)j/4+1/4} \hat{f}(\xi(\eta))[\chi_{\mathcal{C}_{(h)}}(\xi(\eta))U(h), j(\eta) \\ &+ \chi_{\mathcal{C}_{(v)}}(\xi(\eta))U_{(v),j}(\eta)] e^{2\pi i \eta^\top m} d\eta \Big|^2 \\ &= \int_{Q^2} 2^{(2+\alpha_j)j/2+1/2} |\hat{f}(\xi(\eta))|^2 \Big| \xi_{\mathcal{C}_{(h)}}(\xi(\eta))U_{(h),j}(\eta) + \xi_{\mathcal{C}_{(v)}}(\xi(\eta))U_{(v),j}(\eta)\Big|^2 d\eta \\ &= \int_{\mathcal{C}_{(h)}} |\hat{f}(\xi)|^2 |W(2^{-j}\xi)|^2 |v(k_j(\xi_2/\xi_1 - 1))|^2 \\ &+ \int_{\mathcal{C}_{(v)}} |\hat{f}(\xi)|^2 |W(2^{-j}\xi)|^2 |v(k_j(\xi_1/\xi_2 - 1))|^2 d\xi. \end{aligned}$$

Similarly we can get

$$\begin{aligned} \sum_{k\in\mathbb{Z}^2} |\langle f, \psi^{\alpha_j}_{j,k_j,m}\rangle|^2 = &\int_{\mathcal{C}_{(h)}} |\hat{f}(\xi)|^2 |W(2^{-j}\xi)|^2 |v(k_j(\xi_2/\xi_1 + 1))|^2 d\xi \\ &+ \int_{\mathcal{C}_{(v)}} |\hat{f}(\xi)|^2 |W(2^{-j}\xi)|^2 |v(k_j(\xi_1/\xi_1 + 1))|^2 d\xi \end{aligned}$$

which finishes the first case.

**Case 2** Boundary shearlets with $j = 0$: Since supp$W \subset Q^2$, we obtain

$$\begin{aligned}
&\sum_{k\in\mathbb{Z}^2} |\langle f, \psi^{\alpha_j}_{0,\pm 1,m}\rangle|^2 = \sum_{k\in\mathbb{Z}^2} |\langle \hat{f}, \hat{\psi}^{\alpha_j}_{0,\pm 1,k}\rangle|^2 \\
&= \sum_{k\in\mathbb{Z}^2} \left| \int_{Q^2} \hat{f}(\xi) W(\xi) \left[ \chi_{\mathcal{C}_{(h)}} v\left(\frac{\xi_2}{\xi_1} \mp 1\right) + \chi_{\mathcal{C}_{(v)}}(\xi) \left(\frac{\xi_1}{\xi_2} \mp 1\right)\right] e^{2\pi i \xi^\top m} d\xi \right|^2 \\
&= \int_{\mathcal{C}_{(h)}} |\hat{f}(\xi)|^2 |W(\xi)|^2 \left| v\left(\frac{\xi_2}{\xi_1} \mp 1\right)\right|^2 d\xi + \int_{\mathcal{C}_{(v)}} |\hat{f}(\xi)|^2 |W(\xi)|^2 \left| v\left(\frac{\xi_1}{\xi_2} \mp 1\right)\right|^2 d\xi
\end{aligned}$$

**Case 3** Interior shearlets: The substitution $\eta^\top = \xi^\top A_{-j,\alpha_j,(\iota)} S_{-k,(\iota)}$, for $|k| < k_j$, $\iota \in \{h, v\}$ yields

$$\begin{aligned}
\sum_{k\in\mathbb{Z}^2} |\langle f, \psi^{\alpha_j,(h)}_{j,k,m}\rangle|^2 &= \int_{\mathbb{R}^2} |\hat{f}(\xi)|^2 |W(2^{-j}\xi)|^2 \left| v\left(k_j \frac{\xi_2}{\xi_1} - k\right)\right|^2 d\xi, \\
\sum_{k\in\mathbb{Z}^2} |\langle f, \psi^{\alpha_j,(v)}_{j,k,m}\rangle|^2 &= \int_{\mathbb{R}^2} |\hat{f}(\xi)|^2 |W(2^{-j}\xi)|^2 \left| v\left(k_j \frac{\xi_1}{\xi_2} - k\right)\right|^2 d\xi,
\end{aligned}$$

**Case 4** Coarse scaling functions: Since supp$\Phi \subset Q^2$, we have

$$\sum_{k\in\mathbb{Z}^2} |\langle f, \psi_{-1,m}\rangle|^2 = \sum_{k\in\mathbb{Z}^2} \left| \int_{Q^2} \hat{f}(\xi) \hat{\Phi}(\xi) e^{-2\pi i \xi^\top m} d\xi \right|^2 = \int_{\mathbb{R}^2} |\hat{f}(\xi)|^2 |\hat{\Phi}(\xi)|^2 d\xi$$

Finally, using **Case 1** to **Case 2** we can conclude that

$$
\begin{aligned}
&\sum_{\psi\in\mathcal{SH}(\phi,v,(\alpha_j)_j)} |\langle f,\psi\rangle|^2 \\
&= \sum_{k\in\mathbb{Z}^2}\left(|\langle f,\psi_{-1,m}\rangle|^2 + \sum_{\iota\in\{h,v\}}\sum_{j\in\mathbb{N}_0}\sum_{|k|<k_j}|\langle f,\psi^{\alpha_j,(\iota)}_{j,k,m}\rangle|^2 + \sum_{j\in\mathbb{N}_0}\sum_{k=\pm k_j}|\langle f,\psi^{\alpha_j}_{j,k,m}\rangle|^2\right) \\
&= \int_{\mathbb{R}^2}|\hat{f}(\xi)|^2|\hat{\Phi}(\xi)|^2 d\xi \\
&+ \int_{\mathbb{R}^2}|\hat{f}(\xi)|^2 \sum j\in\mathbb{N}_0|W(2^{-j}\xi)|^2\left[\sum_{|k|<k_j}\left|v\left(k_j\frac{\xi_2}{\xi_1}-k\right)\right|^2 + \sum_{|k|<k_j}\left|v\left(k_j\frac{\xi_1}{\xi_2}-k\right)\right|^2\right]d\xi \\
&+ \int_{\mathbb{C}_{(h)}}|\hat{f}(\xi)|^2\sum_{j\in\mathbb{N}_0}|W(2^{-j}\xi)|^2|v(k_j(\xi_2/\xi_1-1))|^2 d\xi \\
&+ \int_{\mathbb{C}_{(v)}}|\hat{f}(\xi)|^2\sum_{j\in\mathbb{N}_0}|W(2^{-j}\xi)|^2|v(k_j(\xi_1/\xi_1-1))|^2 d\xi \\
&+ \int_{\mathbb{C}_{(h)}}|\hat{f}(\xi)|^2\sum_{j\in\mathbb{N}_0}|W(2^{-j}\xi)|^2|v(k_j(\xi_2/\xi_1+1))|^2 d\xi \\
&+ \int_{\mathbb{C}_{(v)}}|\hat{f}(\xi)|^2\sum_{j\in\mathbb{N}_0}|W(2^{-j}\xi)|^2|v(k_j(\xi_1/\xi_1+1))|^2 d\xi \\
&= \int_{\mathbb{R}^2}|\hat{f}(\xi)|^2|\hat{\Phi}(\xi)|^2 d\xi + \int_{\mathbb{R}^2}|\hat{f}(\xi)|^2\sum_{j\in\mathbb{N}_0}|W(2^{-j}\xi)|^2\times \\
&\quad\times\left[\chi_{\mathcal{C}_{(h)}}(\xi)\sum_{|k|\le k_j}\left|v\left(k_j\frac{\xi_2}{\xi_1}-k\right)\right|^2 + \chi_{\mathcal{C}_{(v)}}(\xi)\sum_{|k|\le k_j}\left|v\left(k_j\frac{\xi_1}{\xi_2}-k\right)\right|^2\right]d\xi \\
&\underset{3.15}{=} \int_{\mathbb{R}^2}|\hat{f}(\xi)|^2\left[|\hat{\Phi}(\xi)|^2 + \sum_{j\in\mathbb{N}_0}|W(2^{-j}\xi)|^2\right]d\xi = ||\hat{f}||^2_{L^2(\mathbb{R}^2)} = ||f||^2_{L^2(\mathbb{R}^2)}
\end{aligned}
$$

Then we have the proof finished. This proof was based on the proof of Theorem 3.4 on [2], with the necessary modifications.

□

Now that we know that the general shearlet system forms a frame, we can introduce a simpler idea with $\alpha$ fixed, the so called $\alpha$-Shearlets.

**Definition 3.11** ($\alpha$-Shearlets)**.** For $\phi,\psi,\tilde{\psi}\in L^2(\mathbb{R}^2)$, $\alpha\in(-\infty,2)$ and $c=(c_1,c_2)\in(\mathbb{R}^+)^2$, the *$\alpha$-shearlets system* $\mathcal{SH}(\phi,\psi,\tilde{\psi};\alpha,c)$ is defined as

$$\mathcal{SH}(\phi,\psi,\tilde{\psi};\alpha) = \Phi(\phi;c_1)\cup\Psi(\psi;\alpha,c)\cup\tilde{\Psi}(\tilde{\psi};\alpha,c),$$

where

$$
\begin{aligned}
\Phi(\phi;c_1) &= \{\phi_m = \phi(\cdot - c_1 m) : m\in\mathbb{Z}^2\} \\
\Psi(\psi;\alpha,c) &:= \{\psi_{j,k,m} = 2^{(\alpha+1)j/4}\psi(S_{k,(h)}A_{j,\alpha,(h)}\cdot - M_{c,(h)}m) : j\ge 0, |k|\le\lceil 2^{(\alpha-1)j/2}\rceil, m\in\mathbb{Z}^2\}, \\
\tilde{\Psi}(\tilde{\psi};\alpha,c) &:= \{\tilde{\psi}_{j,k,m} = 2^{(\alpha+1)j/4}\tilde{\psi}(S_{k,(v)}A_{j,\alpha,(v)}\cdot - M_{c,(v)}m) : j\ge 0, |k|\le\lceil 2^{(\alpha-1)j/2}\rceil, m\in\mathbb{Z}^2\},
\end{aligned}
$$

where $M_{c,(h)} = \mathrm{diag}(c_1,c_2)$ and $M_{c,(v)} = \mathrm{diag}(c_2,c_1)$.

Let $\alpha \in (-\infty, 2)$, and recall the form of the scaling sequences set $A_j$ from definition 3.9, we choose $(\alpha_j)_j$ then to be the best possible approximation of $\alpha$,

$$\alpha_j := \underset{\tilde{\alpha}_j \in A_j}{\operatorname{argmin}} |\tilde{\alpha}_j - \alpha|, \quad j \geq 1.$$

It is easy to verify that $2^{\alpha_j j/2} \in \Theta(2^{\alpha_j/2})$ as $j \longrightarrow \infty$. Then the corresponding universal shearlet system (or universal-scaling shearlet system) $\mathcal{SH}(\phi, v, (\alpha_j)_j)$ has asymptotically the same scaling behavior as $\alpha$-shearlets; for further analysis of $\alpha$-shearlets and universal shearlet system we recommend to check [2] and [55]. This generalized theory is very useful in the case that we want to have more freedom on the level of anisotropy of the features of some signal we want to analyze; this also will give us a general framework of sparse representation of different type of features in natural images and videos, the sparse character of the representations let us use the shearlet transform for typical image processing tasks as denoising and inpainting; the latter plays an important role in the light field recovery method presented in this thesis and therefore we will explore it in more detail on the next section.

## 3.4 Image inpainting using Shearlet Parseval Frames

On the last chapter we explained how one can optimize the light field acquisition by using sparse acquisition setups that reduces the number of images of the scene (sampling rate) that one needs to acquire and still be able to recover the light field, this also reduces the complexity of the EPIs acquisition algorithm but adds a new problem; the EPIs obtained by this sparse acquisition technique are as well sparse so one will truncated straight lines instead of straight lines (see Figure 3.12).

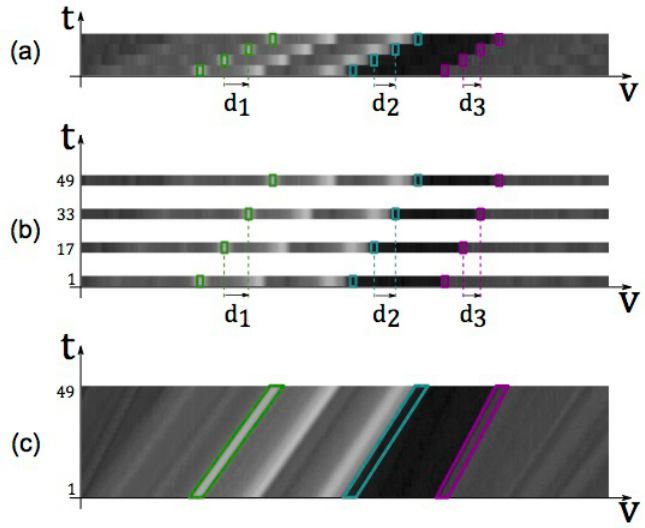



FIGURE 3.12: (a) EPI for a coarsely sampled light field; (b) Correspondeing partially defined densely sampled EPI; (c) Ground truth densely sampled EPI, can be obtaied by inpainting. Figure taken from [3] pp. 6

One would like to recover the lost sections of the EPIs, this task is present in different applications on data processing, since frequently the technologies of data acquisition fails resulting on missing traces; in imaging science, this task is referred as *inpainting* by the similiraty of the physical task of restoring a painting.

There are different algorithmical approaches to the inpainting task; the algorithms based on variational approaches which propagate information from the boundaries and try to guarantee smoothness (see [56]). Another approach is based on applied harmonic analysis combined with ideas of compressed sensing, assuming that certain representation systems provide sparse approximations of the original image, most of the approaches assume that one knows the position and shape of the lost areas (referred as masks).

In this thesis we will follow the second approach exploiding the sparse representation characteristics of the shearlet systems to inpaint epipolar plane images obtained from a sparsely sampled light field; using a combination of harmonic analysis and also a $\ell^1$ minimization technique widely used in compressed sensing.

Before explaining the particular case of inpainting algorithm that we will work with it is worthwhile to present a general abstract framework of inpainting to grow some intuition. For this we need a separable Hilbert space $\mathcal{H}$. Let $x^0 \in \mathcal{H}$ the undamaged (original) *signal* that we want to recover. We will also assume that $\mathcal{H}$ decomposes into an orthogonal sum of two closed subspaces, that we will call the *known part* and the *missing part*,

$$\mathcal{H} = \mathcal{H}_K \oplus \mathcal{H}_M = P_K\mathcal{H} \oplus P_M\mathcal{H}$$

where $P_K$ and $P_M$ are the orthogonal projection operator to the respective space. The inpainting of $x^0$ will be then translated to: "Given a corrupt signal $P_K x^0$, recover the missing part $P_M x^0$.

In the case of image inpainting, we will consider a continuous image model, where $\mathcal{H} = L^2(\mathbb{R}^2)$, where the missing space is $H_M = L^2(\mathcal{M})$ for some measurable set $\mathcal{M} \subset \mathbb{R}^2$, seen as a mask covering the corrupted parts of the painting. In the next we will make further assumptions on the signal and missing space.

We will assume that the signal $x^0$ can be efficiently represented by some Parseval frame $\Phi = (\phi_i)_{i\in I}$ for $\mathcal{H}$, at practical applications this assumption is reasonable since we already mentioned that natural images represented by cartoon-like functions are optimally represented by directional systems as shearlets or curvelets. In the classical theory of sparse representations, this is translated as asking for the solution of the $\ell^0$-minimization problem

$$\min_{c\in\ell^2(I)} ||c||_{\ell^0(I)} \quad \text{subject to} \quad x^0 = T_\Phi^* c = \sum_{i\in I} c_i\phi_i \tag{3.25}$$

where the operator $T_\Phi^*$ is the so-called synthesis operator associated to the frame, then equation 3.25 tries to find a synthesis sequence of $x^0$ that has as few as possible non-zero elements measured by $||c||_0 := \#\{c_i | c_i \neq 0\}$, by using the least possible of Shearlet Coefficients one smooth out the masked image covering then the holes get smoothly covered. This minimization problem is not convex which represents a big complexity issue in the solution method; we will then follow a new strategy called *analysis approach*.

In the *analysis approach* we will consider *analysis coefficients* given by the *analysis operator* $T_\Phi x^0 = (\langle x^0, \phi_i\rangle)_{i\in I}$, since the frame is assumed to obey the Parseval property, we can perform the reconstruction by $x^0 = T_\Phi^*(\langle x^0, \phi_i\rangle)_{i\in I} = \sum_{i\in I}\langle x^0, \phi\rangle\phi_i$. $\Phi$ is not necessarily a basis, $T_\Phi x^0$ might not be a solution of 3.25, so is not yet clear what does

it mean to give a sparse representation in the sense of the analysis approach; to make sense of this we need to use some ideas of *compressed sensing*.

To be able to use the *analysis approach* we will introduce the next $\ell^1$-minimization algorithm:

---

**Algorithm 1:** Inpainting via $\ell^1$-minimization

---

**Input** : Corrupted signal $P_K x^0 \in \mathcal{H}_K$, Parseval frame $\Phi = (\phi_i)_{i\in I}$ for $\mathcal{H}$

**Compute**:

$$x^* = \underset{x\in\mathcal{H}}{\operatorname{argmin}} ||T_\Phi x||_{\ell^1(I)} \quad \text{subject to} \quad P_K x^0 = P_K x \qquad (\ell^1 - \text{INP})$$

**Output** : recovered signal $x^* \in \mathcal{H}$

---

In other words, Algorithm 1 minimizes the $\ell^1$-norm among all possible reconstruction candidates, which is the set of all signals coinciding with $x^0$ on $\mathcal{H}_K$. In the case where the undamaged signal $x^0$ is sufficiently sparsified by $\Phi$, and then it has a small $\ell^1$-norm of $T_\Phi x^0$, the solution $x^*$ will provide a good reconstruction of the important features on $x^0$.

We would like to prove some error estimates for the recovery by Algorithm 1, for this, we will introduce some analysis tools that will also give us further insight about the structure of the proposed abstract model.

**Definition 3.12** ($\delta$-cluster sparsity)**.** Let $\delta > 0$ and $\Gamma \subset I$. A signal $x \in \mathcal{H}$ is called $\delta$-clustered sparse in $\Phi$ with respect to $\Gamma$, if

$$||\mathbf{1}_{\Gamma^c} T_\Phi x||_{\ell^1} \leq \delta \tag{3.26}$$

In this case, $\Gamma$ is said to be $\delta$-cluster for $x$ in $\Phi$.

A signal will be $\delta$-clustered sparse if the analysis coefficients are highly concentrated on $\Gamma$. This definition will only have a useful meaning when $x$ is $\delta$-clustered sparse with respect to a small cluster $\Gamma$, and $\delta$ sufficiently small at the same time. Independently we can define the concept of *cluster coherence*.

**Definition 3.13** (Cluster coherence)**.** Let $\Gamma \subset I$, the cluster coherence of $\Phi$ with respect to $\mathcal{H}_M$ and $\Gamma$ is given by

$$\mu_c(\Gamma, P_M\Phi) := \max_{j\in I} \sum_{i\in\Gamma} |\langle P_M\phi_i, P_M\phi_j\rangle|,$$

where $P_M\Phi := (P_M\phi_i)_{i\in I}$.

The cluster coherence can be understood as an abstract measure for the gap size; we do not know $P_M x^0$, so we would like to know the maximal amount of missing information. The Parseval expansion $x^0 = \sum_{j\in I}\langle x^0, \phi_j\rangle\phi_j$ help us to estimate the analysis coefficientes of its projection onto $\mathcal{H}_M$:

$$|\langle P_M x^0, \phi_i\rangle| \leq \sum_{i\in I} |\langle x^0, \phi_j\rangle||\langle P_M\phi_j, \phi_i\rangle| = \sum_{i\in I} |\langle x^0, \phi_j\rangle||\langle P_M\phi_j, P_M\phi_i\rangle|. \tag{3.27}$$

The scalar products $|\langle P_M\phi_j, P_M\phi_i\rangle|$ broadly indicate how "correlated" the measurments with $\phi_i$ and $\phi_j$ are on $\mathcal{H}_M$. To compute the recovery error, it was proposed before (see [57]) to estimate the ditance between $x^0$ and its $\ell^1$-recovery $x^*$ with the Hilbert norm. This worked well to estimate an upper bound, but to show an optimality result (lower bound), measuring $||\cdot||$ is not useful, since the Parseval property of $\Phi$ gives $||x^0 - x^*|| = ||T_\Phi(x^0 - x^*)||_{\ell^2}$, the $\ell^2$-norm in comparison with the $\ell^1$ has an averaging effect on the coefficientes, then some sparsity features might be hidden when using the Hilbert space norm. We will proposed a new norm to measure the error, the $\ell^1$-analysis norm.

**Definition 3.14** ($\ell^1$-analysis norm)**.** Let $x \in \mathcal{H}$ and $\Phi$ be a Parseval frame for $\mathcal{H}$. Then we define the $\ell^1$-analysis norm, with respect to $\Phi$, by

$$||x||_{1,\Phi} := ||T_\Phi x||_{\ell^1} = ||(\langle x, \phi_i\rangle)_{i\in I}||_{\ell^1}$$

and the $\ell^1$-analysis space given by

$$\mathcal{H}_{1,\Phi} := \{x \in \mathcal{H} : ||x||_{1,\Phi} < \infty\}$$

equipped with $||\cdot||_{1,\Phi}$

Since $\Phi$ forms a Parseval frame, its analysis operator $T_\Phi$ is injective, and therefore the tuple $(\mathcal{H}_{1,\Phi}, ||\cdot||_{1,\Phi})$ defines a normed vector space, moreover

$$||x|| = ||T_\Phi x||_{\ell^2} \leq ||T_\Phi x||_{\ell^1} = ||x||_{1,\Phi} \tag{3.28}$$

then we have the embedding $\mathcal{H}_{1,\Phi} \hookrightarrow \mathcal{H}$, then, by picking $x = x^* - x^0$, we can conclude that small recovery errors in $\mathcal{H}_{1,\Phi}$ always imply small ones in $\mathcal{H}$. With this tools we can provide a estimate of the recovery error.

**Theorem 3.9.** Let $\delta > 0$ and $\Gamma \subset I$ be a $\delta$-cluster for $x^0$ in $\Phi$. Moreover, assume that $\mu_c(\Gamma, P_M\Phi) < 1/2$. If $x^0 \in \mathcal{H}_{1,\Phi}$ then the $\ell^1$-minimizer $x^*$ of the Algorithm 1 is also contained in $\mathcal{H}_{1,\Phi}$ and satisfies

$$||x^* - x^0||_{1,\Phi} \leq \frac{2\delta}{1 - 2\mu_c(\Gamma, P_M\Phi)} \tag{3.29}$$

*Proof.* The proof can be found in [57]. □

The Theorem 3.9 gives good estimates if $\Phi$ sparsifies $x^0$ and the size of the missing space, the latter measured in terms of cluster coherence is not too large. We would like to pick a cluster $\Gamma$ such that $x^0$ becomes $\delta$-clustered sparse for small $\delta$ and at the same time $\mu_c$ shall not exceed the bound of $1/2$; even the concepts if cluster sparsity and cluster coherence were introduced independently, both notions are now joined together in terms of $\Gamma$; enlarging $\Gamma$ makes the Equation 3.29 true for smalle $\delta$, while $\mu_c(\Gamma, P_M\Phi)$ increases and could exceed $1/2$, this tell us that applying the Theorem 3.9 involves actually the fundamental task of choosing an appropriate cluster (see [57] for a more detailed analysis).

So far we have a general abstract framework for inpainting for Parseval frames, we would like to have an appropriate parseval frame to inpaint natural images (in the case of this thesis Epipolar Plane Images), in the sense that the recovery error is small; we have different options, for instance Universal Shearlet Systems and Wavelet Systems. Theorem 3.8 we know that for a scaling sequence $(\alpha_j)_j$ the related Universal Shearlet System $\mathcal{SH}(\phi, v, (\alpha_j)_j)$ form a Parseval frame for $L^2(\mathbb{R}^2)$ and that they behave asymptotically equally as the $\alpha$-Shearlets system $\mathcal{SH}(\phi, \psi, \tilde{\psi}; \alpha)$, then the $\alpha$-Shearlets and the Universal Shearlet Transform can be used in the inpainting framework. In the case of the Wavelet Systems, they will also form a frame for $L^2(\mathbb{R}^2)$ under some assumptions on the mother wavelet function, the scaling and the translation parameter (see [43] pp. 202).

In order to compare the shearlets and the wavelets performance for inpaiting, we want to present a benchmark that is related to the principal problem of the thesis, that is the inpainting of sparse sampled epipolar plane images (see Figure 3.12). Cartoon-like images are we know are governed by edges, the image to be inpainted as a benchmark is a masked linear singularity that can be viewed as a strip; epipolar plane images corresponding to sparse sampled light fields have also strips as missing space so this example will be enough. Many details in the analysis will be ommited but they are well explained in [2].

As we cannot work on actual edges in $L^2(\mathbb{R}^2)$ and other Lebesgue spaces since they have measure zero, we will work with a line-distribution given by $\omega\mathcal{L}$ acting on Schwartz functions $f \in \mathbb{S}(\mathbb{R}^2)$, i.e. $\omega\mathcal{L} \in \mathbb{S}'(\mathbb{R}^2)$, if supp $\omega \subset [-\rho, \rho]$ then its action will be defined by

$$\langle \omega\mathcal{L}, g\rangle = \int_{-\rho}^{\rho} \omega(x_1) g(x_1, 0) dx_1, \quad \forall g \in \mathcal{S}(\mathbb{R}^2)$$

where $\omega$ is a smooth weight and $\rho > 0$. The weight $\omega$ sets up the linear singularity that is smooth in the vertical direction, while the value of $\rho$ corresponds to the length of the singularity. We will mask the linear singularity (weighted distribution) $\omega\mathcal{L}$ with the mask

$$M_h = \{(x_1, x_2) \in \mathbb{R}^2 : |x_1| \leq h\}, \quad h > 0$$

The observed signal then is

$$f = \mathbf{1}_{\mathbb{R}^2 \setminus M_h} \cdot \omega\mathcal{L}$$

see Figure 3.13. We decompose $\omega\mathcal{L}$ by the same subbands $F_j$, $\omega\mathcal{L} \mapsto \omega\mathcal{L}_j = \omega\mathcal{L} * F_j$, denote $h = h_j$ and set

$$f_j = \mathbf{1}_{\mathbb{R}^2 \setminus M_{h_j}} \cdot \omega\mathcal{L}_j$$

Let $\{\psi_\lambda\}$ denote a particular wavelet Parseval frame and $\{\sigma_\eta\}_\eta$ a particular shearlet Parseval frame (could be, classical shearlets, cone-adapted shearlets, $\alpha$-shearlets or universal shearlets). Then we can rewrite the minimization problem $\ell^1 - \text{INP}$ as

$$W_j = \underset{\tilde{W}_j}{\text{argmin}} ||(\langle \tilde{W}_j, \psi_\lambda \rangle)_\lambda||_{\ell^1} \quad \text{subject to} \quad f_j = \mathbf{1}_{\mathbb{R}^2 \setminus M_{h_j}} \cdot \tilde{W}_j$$

for wavelet-based inpainting and

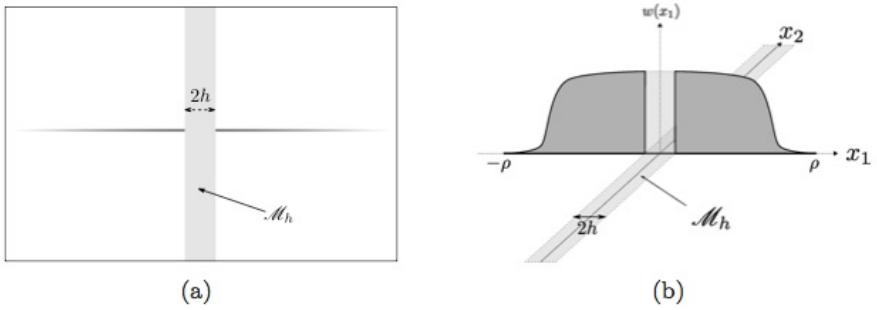



FIGURE 3.13: (a) Sketch of the corrupted modeling image, the line singularity has no "thickness" but some (gray-scale) intensity. (b) "Graph" of the corrupted line distribution $P_K\omega\mathcal{L}$ which is compactly supported on the $x_1$-axis. Figure taken from [2] pp. 15

$$S_j = \underset{\tilde{S}_j}{\operatorname{argmin}} ||(\langle \tilde{S}_j, \sigma_\eta \rangle)_\eta||_{\ell^1} \quad \text{subject to} \quad f_j = \mathbf{1}_{\mathcal{R}^2 \setminus M_{h_j}} \cdot \tilde{S}_j$$

for the shearlet-based inpainting. Now we can state a Theorem that compares both systems in the inpainting task.

**Theorem 3.10.** For $h_j = o(2^{-j})$ (the critical thresholding case) as $j \longrightarrow \infty$,

$$\frac{||W_j - \omega\mathcal{L}_j||_2}{||\omega\mathcal{L}_j||_2} \longrightarrow 0, \quad j \longrightarrow \infty$$

For $h_j = o(2^{-j/2})$ as $j \longrightarrow \infty$,

$$\frac{||S_j - \omega\mathcal{L}_k||_2}{||\omega\mathcal{L}_j||_2} \longrightarrow 0, \quad j \longrightarrow \infty$$

*Proof.* Due the technical level of the proof, we will refer to [59] for the detailed proof. □

The Theorem 3.10 shows that Shearlets is a better choice than Wavelets as a Sparsifying Parseval Frame to perform the inpainting of natural images, in particular of EPIs, therefore is the system that we picked in this thesis.

There are different methods to minimize the $\ell^1$-norm in the inpainting algorithm 1, we will use the iterative hard thresholding method since is the one that is traditionally used due to its effectiveness, the implementation and application of this algorithm for our task of EPIs-inpatinting will be explained in the Chapter 4.

## 3.5 0-Shearlets

In the last section we introduce a generalization of the classical cone-adpated shearlet transform allowing a more flexible scaling matrix $A_{j,\alpha_j,(\iota)}$, where $\alpha_j \in (-\infty, 2)$ changes the "level" of anisotropy of the features the transform is sensible to. We also know by Theorem 3.8 that Shearlet System generated by the sequence of scaling parameters $(\alpha_j)_{j\in\mathbb{N}_0}$ will form a frame if the sequence form a scaling sequence, i.e. $\alpha_j$ is a multiple of $2/j$.

In the case when $\alpha_j = 2$ for all $j = 2$ we have a isotropic scaling case, where isotropic features like circles are well represented; it is the case of the wavelets. In the other hand when $\alpha_j = 1$ for all $j$, then we have the classical cone-adapted shearlet system and the related parabolic scaling is well suited to approximate functions with singularities over parabolic curves.

As we already mentioned several times, the final task of this thesis is to inpaint EPIs form sparse sampled light fields using Shearlets. As we can see in Figure 3.12 Epipolar Plane Images consist of straight-line structures, so one would like to have a scaling operation suited to straight-lines; this is performed using the sequence of scaling parameters given by

$$\alpha_j = -\frac{2}{j} \in A_j, \quad \forall j \in \mathbb{Z}$$

therefore the related universal shearlet system will form a Parseval frame and our inpainting framework is valid. By choosing like in this case very small $\alpha_j$ one produces more anisotropic elements which will approximate properly straight-line formed structures.

As we saw in Section 3.1, the generating functions of a cone-adapted separable compactly supported shearlet system are given by:

$$\begin{aligned}\psi(x_1, x_2) &= \psi_1(x_1)\phi_1(x_2)\\ \phi(x_1, x_2) &= \phi_1(x_1)\phi_1(x_2)\\ \tilde{\psi}(x_1, x_2) &= \psi(x_2, x_1)\end{aligned}$$

where $\phi_1$ and $\psi_1$ are 1D scaling and wavelet functions. To obtain a smaller overlaping of the elements in the fourier transform one can multiply the separable generator by a 2D directional fan filter

$$\hat{\psi}^{\text{nonsep}} = P(\xi_1/2, \xi_2)\hat{\psi}_1(\xi_1)\hat{\phi}_1(\xi_2)$$

If one considers a multiresolution analysis with wavelet and scaling function $\psi_1, \phi_1 \in L^2(\mathbb{R}^2)$ given by

$$\begin{aligned}\phi_1(x_1) &= \sum_{n_1 \in \mathbb{Z}} h(n_1)\sqrt{2}\phi_1(2x_1 - n_1)\\ \psi_1(x_1) &= \sum_{n_1 \in \mathbb{Z}} g(n_1)\sqrt{2}\phi_1(2x_1 - n_1)\end{aligned}$$

The universal shearlet system generated by the scaling sequence $(\alpha_j)_{j\in\mathbb{Z}} = (-2/j)_{j\in\mathbb{Z}}$ will be referred as $0 - \textit{Shearlets}$ since $\alpha_j \underset{j\to\infty}{\longrightarrow} 0$. The scaling matrix related to the 0-shearlets will be given by

$$A_j = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{-1} \end{pmatrix}$$

The choice as expected will provide scaling only by one axis and the shearing will change the direction of the scaling. Let $J \in \mathbb{N}$ the maximum scale, then the shearlet system for $\Psi(\psi)$ is formed by the functions

$$\Psi(c, \psi) = \psi_{j,k,m}, \quad |k| \leq 2^{j+1}, \quad j = 0, \ldots, J-1,$$

where

$$\psi_{j,k,m}(x) = 2^{j/2}\psi(S_k A_j x - M_{c_j} m), \tag{3.30}$$

and $c_j = (c_1^j, c_2^j)$ are sampling constants for translation. It is easy to see that

$$\psi_{j,k,m}(x) = \psi_{j,0,m}\left(S_{\frac{k}{2^j+1}} x\right). \tag{3.31}$$

Following the procedure of [51], it can be shown that the digital filter corresponding to $\psi_{j,0,m}$ is given by

$$\psi_{j,0}^d(m) = (p_j * (g_{J-j} \otimes h_{J+1}))(m), \tag{3.32}$$

where $\otimes$ denote the tensor product such that

$$(g_{J-j} \otimes h_{J+1})(m) = g_{J-j}(m_1) h_{J+1}(m_2),$$

and $\{p_j(n)\}_{n\in\mathbb{Z}}$ are the Fourier coefficients of the trigonometric polynomial $P(2^{J-j-1}\xi_1, 2^{J+1}\xi_1)$, $\{h_j(n)\}_{n\in\mathbb{Z}}$ and $\{g_j(n)\}_{n\in\mathbb{Z}}$ are the Fourier coefficients of the respective trigonometric polynomial

$$\hat{h}_j(\xi) = \prod_{k=0,\dots,j-1} \hat{h}(2^k\xi),$$
$$\hat{g}_j(\xi) = \hat{g}(2^{j-1}\xi)\hat{h}_{j-1}(\xi)$$

and $\hat{h}_0 \equiv 1$, one can see in Figure 3.14 the frequency responses of this digital filters until scale $j = 4$.

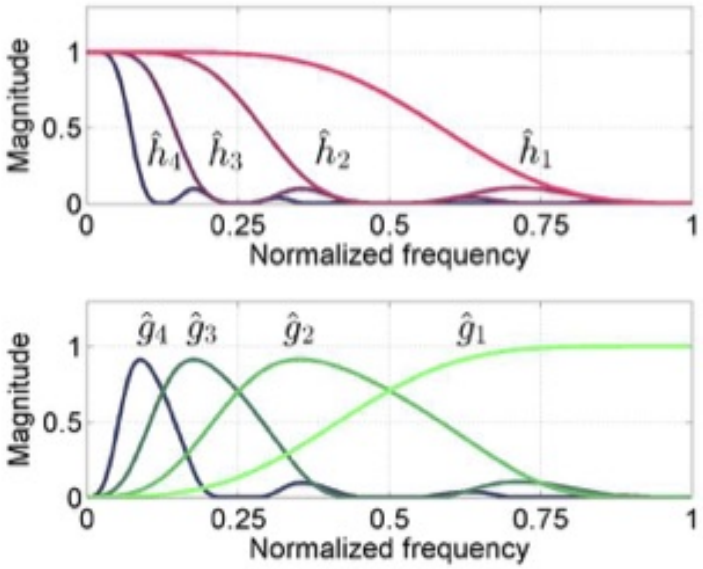



FIGURE 3.14: Frequency responses of the scaling and wavelet filters $h_j, g_j, j = 1, \dots, 4$. Figure taken from [3] pp. 4

As it was presented for first time in [52], the shearing transform $S_{k2^{-j}}$, $j \in \mathbb{N}$, $k \in \mathbb{Z}$ does not preserve the regular grit $\mathbb{Z}^2$, therefore its digitalization is non-trivial. To tackle this issue, one needs to refine the $\mathbb{Z}^2$ grid along the $x_1 - axis$ by a factor $2^j$, then the new grid $2^{-j}\mathbb{Z} \times \mathbb{Z}$ is invariant under the $S_{k2^{-j}}$ transform.

For an arbitrary $r \in \ell^2(\mathbb{Z}^2)$, the shear transform $S_{k2^{-j}}$ can be implemented as a digital filter

$$S^d_{k2^{-j}}(r) = ((2^j r_{\uparrow 2^j} *_1 \tau_j)(S_k \cdot) *_1 \overline{\tau}_j)_{\downarrow 2^j} \tag{3.33}$$

where $\tau_j$ is a digital low-pass filter with normalized cutoff frequency at $2^{-j}$.

Using the Equations 3.30, 3.31, 3.32 and 3.33 and the proper choice of $c_j$ it can be shown that the digital filter corresponding to $\psi_{j,k,m}$ is given by

$$\psi^d_{j,k}(m) = (S^d_{k2^{-(j+1)}}(p_j * g_{J-j} \otimes h_{J+1}))(m).$$

A digital filter corresponding to separable elements of the transform $\phi_m$ related to the scaling function, is given by $\phi^d = (h_J \otimes h_J)(m)$. Then, the discrete shearlet transform associated with the set of elements $\Psi(c;\psi)$ and corresponding to frequency plane region $C_\psi$ is defined as follows

$$\mathcal{DST}_{j,k,m}(f_J) = (f_J * \overline{\psi^d_{j,k}})(m),$$

where $f_J(n)$ for $n \in \mathbb{Z}^2$ are discrete samples of $f \in L^2(\mathbb{R}^2)$, $j = 0, \ldots, J-1$, $|k| \leq 2^{j+1}$, $m \in \mathbb{Z}^2$.

To compute the inverse transform we need to construct the dual frame; in order to do that we first set

$$\hat{\Psi}^d = |\hat{\phi}^d|^2 + \sum_{j=0,\ldots,J-1} \sum_{|k| \leq 2^{j+1}} (|\hat{\psi}^d_{j,k}|^2 + |\hat{\tilde{\psi}}^d_{j,k}|^2).$$

The dual shearlet filters are defined by

$$\hat{\phi}^d = \frac{\hat{\phi}^d}{\hat{\Psi}^d}, \quad \hat{\gamma}^d_{j,k} = \frac{\hat{\psi}^d_{j,k}}{\hat{\Psi}^d}, \quad \hat{\tilde{\gamma}}^d_{j,k} = \frac{\hat{\tilde{\psi}}^d_{j,k}}{\hat{\Psi}^d}$$

One can see an illustration of the obtained system in the frequency plane $\hat{\Psi}^d$ for $J = 2$ in Figure 3.15. Finally, the reconstruction formula is given by

$$f_J = (f_J * \overline{\phi}^d) * \phi^d + \sum_{j,k} (f_J * \overline{\psi}^d_{j,k}) * \gamma^d_{j,k} + \sum_{j,k} (f_J * \overline{\tilde{\psi}}^d_{j,k}) \tilde{\gamma}^d_{j,k}$$

Following the methodology of [3] we will be interested only in the transform elements where the shearing has positive sign, i.e. $0 \leq k \leq 2^{j+1}$, the corresponding transform elements are ccovering the frequency domain region highlighted in Figure 3.16. Then, we use the direct transform $S$ for discrete values $f_J$ and $j = 0, \ldots J-1$, $k = 0, \ldots, 2^{j+1}$, $m \in \mathbb{Z}^2$ defined as

$$S(f_J) = \{c_{j,k}(m) = (f_J * \overline{\psi}^d_{j,k})(m), c_0(m) = (f_J * \overline{\phi}^d)(m) : \quad j = 0, \ldots, J-1, k \leq 2^{j+1}, m \in \mathbb{Z}^2\}. \tag{3.34}$$

where $S$ is known as the synthesis operator related to the 0-Shearlets frame. The inverse transform $S^*$ is then given by the so called synthesis operator

$$S^*(\{c_{j,k}, c_0\}) = \sum_{\substack{j=0,\ldots,J-1 \\ k=0,\ldots,2^{j+1}}} (c * \gamma^d_{j,k})(m) + (c_0 * \phi^d)(m). \tag{3.35}$$

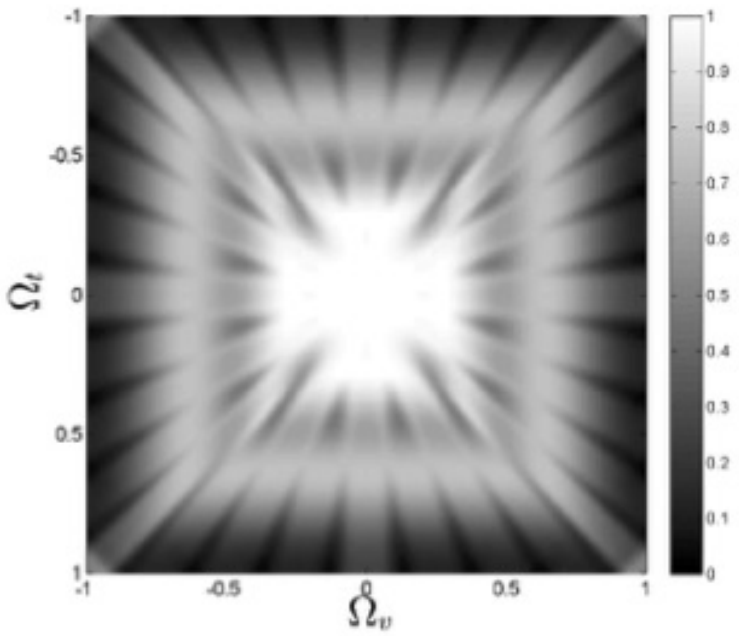



FIGURE 3.15: $\hat{\Psi}^d$ corresponding to constructed shearlet transform for $J = 2$. Figure taken from [3] pp. 4

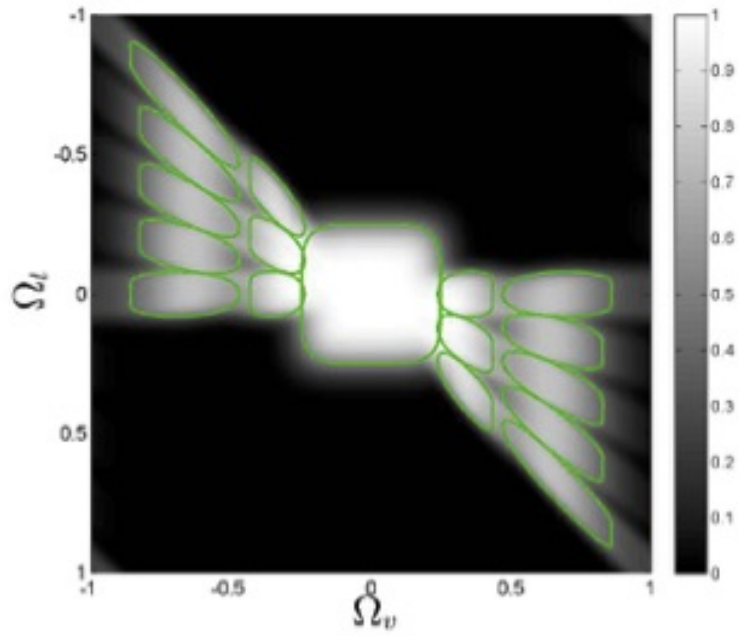



FIGURE 3.16: Frequency domain support of shearlet transform elements used in reconstruction. Figure taken from [3] pp. 4

In the case of the Shearlet Transform implementation that we used for this thesis (Shearlab.jl) the 0-Shearlet transform can be perform by choosing for each scale $j$ the number of shearings to be such that $|k| \leq 2^{j+1}$.

As a final remark for this section it is worth to mention that the 0-Shearlet system is closely related to the Ridgelet Transform (see [**?**]), in the sense that both are computed by an inner product of an admissible generating function with different directional, scaling and translational parameters, where the scaling operation is performed in just one direction; the difference is that ridgelets perform directional operation by rotating the elements instead of shearing.

We have now all the tools necessary to perform the light field reconstruction using the Epipolar Plane Images method with sparsely sampled EPIs; in the next chapter we will present the minimization method used to perform such reconstruction as well as the results obtained numerically and the comparison with other light field reconstruction methods using different softwares and procedures.

# Chapter 4

# Inpainting Sparse Sampled Epipolar-plane and Computing Depth Map

We just presented in Subsection 3.4 a general framework for image inpainting using Parseval Frames and we also presented the comparison of general Shearlet Parseval Frames (which include Universal Shearlets, $\alpha$-Shearlets and Cone Adapted Shearlets) and Wavelet Parseval Frames in the task of inpainting a line-distribution singularity, having the conclusion that Shearlets are better by its directional sensitivity.

We also presented in Subsection 3.5 the particular case of Universal Shearlets with scaling sequence given by the parameters $\alpha_j = -2/j$, which generates a Parseval frame that is a good choice for the representation of singularities distributed over straight lines, and therefore is a good option for inpainting Epipolar Plane Images that are formed by linear structures.

In this chapter we will present a particular algorithm that we used to inpaint EPIs related to sparse sampled light fields using 0-Shearlets in the Julia implemenation of Shearlab (Shearlab.jl) which is called Iterative Hard Thresholding; we will also present the results on the inpainting of a particular data set (the Church Data set already mentioned in Chapter 2).

Using the inpainted EPIs we will present a line detection algorithm called Hough Line Transform that allows us to get the slopes of the lines related to different features in in the EPIs and therefore let us compute the depth map of the scene concluding the light field reconstruction task that we were looking for in this thesis.

## 4.1 Iterative thresholding algorithm for EPIs inpainting using 0-Shearlets

To formulate the light field reconstruction algorithm in discrete domain we will assume that that the starting coarse set of views is a downsampled version of the unknown densely sampled light field we are trying to reconstruct. The uniformly distributed cameras imply the possibility of estimating a common upper bound of disparities between consecutive views that we will call $d_{\max}$; as we saw in Subsection 2.5.1 for densely sampled EPI's one need to ensure maximum 1 pixel disparity between nearby views (this in order to avoid aliasing and other artifacts), this representing minimum sampling rate law similar to Nyquist-Shannon sampling theorem. The given sparse set of views are regarded as taken at each $d_{max} = \lceil d_{max} \rceil$-th view of a densely sampled Light Field.

As a very illustrative example one refers to the Figure 3.12 where we have an EPI representation of four views with 16 pix disparity; we also established in Subsection 2.5.1 that it is sufficient to compute the slope of the lines representing certain scene points in the Epipolar plane images to know the relative depth in the scene of the correspondent feature point, this using the equation 2.4; for that we will need to be able to see clear straight lines in the EPIs.

On Figure 3.12 (a) one can see an sparse sampled Epipolar Plane Image, where the lines are not distinguishable; on Figure 3.12(b) when we separate the layers of the sparse sampled Epipolar Plane Image with 16px of disparity between the consecutive views the lines start to form; and finally the lines are clear in Figure 3.12(c), that represents the correspondent densely sampled Epipolar Plane Image which we want to recover by inpainting the sparse version.

To proceed with the mathematics of the optimization problem that performs the inpainting, we will assume that the densely sampled EPI is a square image denoted by $y^* \in \mathbb{R}^{N\times N}$ where $N = md_{max}$ and $m$ is a number of avialable views. Given the samples $y \in \mathbb{R}^{N\times N}$ of the dense $y^*$ obtained by

$$y(i,j) = M(i,j)y^*(i,j) \tag{4.1}$$

where $M \in \mathbb{R}^{N\times N}$ is a measuring matrix from which one can get the mask for the inpainting problem, such that $M(kd_{max}, \cdot) = 1$ for $k = 1, \ldots, m$ and 0 elsewhere. The measurements $y$ form an incomplete EPI where only rows from the available images are presented, while everywhere else EPI values are 0. We can rewrite the Equation 4.1 as $y = Hy^*$ by lexicographically reordering the variables $y, y^* \in \mathbb{R}^{N^2}$, $H \in \mathbb{R}^{N^2\times N^2}$.

Let $\mathcal{SH}(\psi, \phi, (-2/j)_j \in \mathbb{Z})$ be the system of 0-Shearlets, defined in the Subsection 3.5; in addition let $S : \mathbb{R}^{N\times N} \longrightarrow \mathbb{R}^{N\times N\times\eta}$ and $S^* : \mathbb{R}^{N\times N\times\eta} \longrightarrow \mathbb{R}^{N\times N}$ the analysis and synthesis operator related to the 0-Shearlet system defined in equations 3.34 and 3.35, where $\eta$ is the number of all translation invariant transform elements.

The reconstruction problem of $y^*$ defined by the sampling matrix $M$ and the measurements $y$ can be cast as an inpainting problem following the framework of Subsection 3.4 given by,

$$x^* = \underset{x\in\mathbb{R}^{N\times N}}{\operatorname{argmin}} ||S(x)||_1, \quad \text{subject to} \quad y = Mx \tag{4.2}$$

The algorithm that we will use to solve the optimization problem will make use of iterative procedures applied in morphological component analysis approaches, which have been originally proposed for decomposing images into piecewise-smooth and texture parts (see [61] and [62]), this algorithm is also known as Iterative Hard Thresholding. Following the approach of Vagharshakyan [3], we aim to recunstruct the EPI $y^*$ by performing regularization in the shearlet transform domain by its sparsifying properties for cartoon-like functions.

The solution is sought in the form of the algorithm 2.

In the Algorithm 2, the operator $T_\delta$ is the hard thresholding operator given by:

$$(T_\delta x)(k) = \begin{cases} x(k), & |x(k)| \geq \delta \\ 0, & |x(k)| < \delta \end{cases}$$

**Algorithm 2:** Inpainting via iterative hard thresholding

**Input** : Sparse EPI $y$, sampling matrix $M$, $\delta_{init}$,$\delta_{min}$, iterations

**Compute**: $x_0 := 0$;
$\delta_0 := \delta_{init}$;
$\lambda := (\delta_{min})^{1/(iterations-1)}$;
$\Gamma_0 := \text{supp}(S(x_0))$;
$\beta_0 := S_{\Gamma_0}(y - Mx_0)$;
$\alpha_0 = \frac{||\beta_0||_2^2}{||MS^*(\beta_0)||_2^2}$;
**for** $n := 0$ **to** (iterations-1) **do**
$x_{n+1} = S^*(T_{\delta_n}(S(x_n + \alpha_n(y - Mx_n))))$;
$\Gamma_{n+1} := \text{supp}(S(x_{n+1}))$;
$\beta_{n+1} := S_{\Gamma_{n+1}}(y - Mx_{n+1})$;
$\alpha_{n+1} := \frac{||\beta_{n+1}||_2^2}{||MS^*(\beta_{n+1})||}$;
$\delta_{n+1} := \lambda\delta_n$;
**end**

**Output** : Inpainted EPI $x_{iterations}$

The thresholding level $\delta_n$ decreases with the iteration number linearly in the range $[\lambda_{max}, \lambda_{min}]$. The sequence $x_n$ that converges to $x^*$ reaches a solution of the problem 4.2, we refer to Figure 2.17 for the complete pipeline of the reconstruction method.

Here $\alpha_n$ is an acceleration parameter; in the usual inpainting algorithms based on the Shearlet Transform the chosen parameter is $\alpha_n = 1$ (see [58] and [51]) but the convergence in this case is slow and can be accelerated by using $\alpha_n > 1$. It is also not optimal to take alpha too high, since this can cause instability. One can see on Figure 4.1 that convergence speed increases when increasing fixed values $\alpha_n = \alpha$ up to some level where the algorithm starts to diverge.

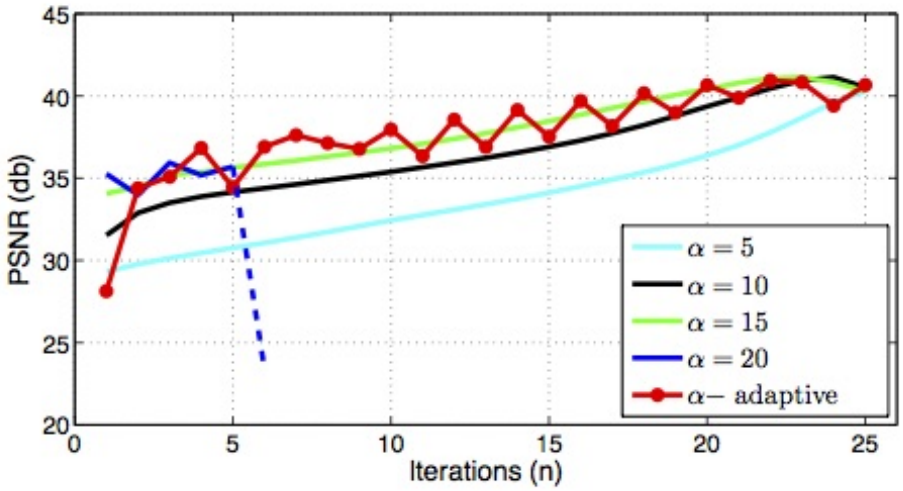



FIGURE 4.1: Reconstruction performance (expressed in terms of Peak Signal-to-Noise radio) dependence on acceleration coefficients $\alpha_n$, for contant value for all iterations $\alpha_n = \alpha$, increasing $\alpha$ brings accelerating convergence, but after some limit, the reconstruction starts to diverge ($\alpha = 20$). Figure taken from [3] pp. 7

The approach that we will use and is presented in Algorithm 2 was proposed by T. Blumensath and M. Davies in 2010 in their article "Normalised Iterative Hard Thresholding; guaranteed stability and performance" [63]. This algorithm applies an iteration-adaptative selection of the parameter given by

$$\alpha_n = \frac{||\beta_n||_2^2}{||MS^*(\beta_n)||_2^2}$$

where $\beta_n = S_{\Gamma_n}(y - Mx_n)$ and $S_{\Gamma_n}$ is the shearlet transform decomposition only for coefficients from $\Gamma_n = \text{supp}(S(x_n))$; it is important to be noticed that $||\cdot||_2$ is the Eucledian norm in the associated $N^2\eta$-dimensional Eucledian space of $\mathbb{R}^{N\times N\times\eta}$ with elements arranged in lexicographic order. The convergence rate of the adaptative selection is also illustrated in Figure 4.1 and one can see that the adaptation provides high convergence speed and stable reconstruction we refer to the original paper [63] for a more detailed analysis of the convergence and stability conditions, which for our case are fulfilled using the fact that the 0-Shearlets system form a Parseval Frame.

As we discurssed in Subsection 3.5 we are not obligated to use all the general shearlet transform atoms, instead we favor the use of atoms which are associated with valid directions in EPI; the support of those atomis is illustrated in Figure 3.16. The scales of the shearlet transform are constructed in dyadic manner, therefore we are choosing $J = \lceil log_2 d_{max} \rceil$ number of scales. In order to perform this programatically using the software Shearlab.jl in every scale we choose $2^{j+1}+1$ shears ($j = 0, \ldots, J-1$) to cover the region.

Finally, in the following subsections we will present the resulting inpainted EPIs of the Church data set, as well as the technique that we use to detect lines in the inpainted EPIs and finally compute the depth map.

## 4.2 Results of sparse EPIs inpaiting

In order to give the full picture of the EPIs inpainting algorithm implementation and results, we will present the set of ideas that made us take some decisions in the way to design the implementation of the inpainting algorithm.

As we presented to Section 2.5 the data set that we used was a series of pictures of a Church, given by the research group of Professor Markus Gross in the Disney Research Center. The data set has 100 different views of the scene, we present in 4.2, 4.3 and 4.4 the 1st, 55th and 100th views of the data set.

- **First step (Track Points):** The first step in the implementation was the design the code to track the points; in the Subsection 2.5.5 we presented the followed algorithm (Lucas-Kanade method with Shi-Tomasi corner detector) whose implementation using OpenCV Python's API is presented in Appendix A with the results were stored in a data frame that one can find in the repository of this thesis (`https://github.com/arsenal9971/MThesis/blob/master/Code_notebooks/church_tracking.csv`) and are presented in Figure 2.28.

- **Second Step (Paint sparse EPIs):**Once we had the data of the tracked points we proceed to paint the EPIs related to fixed $y$-positions in the scenes, for this task we used the julia code presented in Appendix B, where we used horizontal strips

FIGURE 4.2: 1st picture of the Church Data Set

FIGURE 4.3: 55th picture of the Church Data Set

FIGURE 4.4: 100th picture of the Church Data Set

of width $2\epsilon = 16.0$ pixels to captures tracked points in the scene at different fixed $y$ positions and follow them along the different views, we also used the same data set to paint the corresponding Sparse Views with a constant disparity between consecutive views $d_{max} = 7px$; this maximum disparity was chosen by trying different disparities and pick the best one in terms of the painting speed (the greater the disparity, the less time it takes to paint the sparse EPIs) and the Shearlet-based inpainting performance using a fixed number of iterations (50) in the iterative thresholding algorithm; where we measure the performance mostly in terms of how many lines of the original EPI are captured by our line detection algorithm that will be explained in the next subsection. One can see in Figure 4.5 the strip of points in the first image that were tracked to perform the benchmark to compute the best disparity parameter for our task; the Figure 4.6 is the associated densely sampled EPI and the associated sparsely sampled EPIs with different disparities $d_{max}$ are presented in Figure 4.7, 4.8, 4.9 and 4.10.

Finally on Figure 4.11 one can see the different running time obtained for different disparities (clearly the smaller the disparity, the more pictures you need to take and therefore the longest the running time); as we mentioned on Subsection 2.5.5 we used a Macbook Pro with OSX 10.10.5, 8GB of memory, 2.7 GHz Intel Core i5 Processor and Graphic Card Intel Iris Graphics 6100 with 1536 MB of graphic memory (relatively a low end computer system). One can find the whole set of strips, sparse and dense EPIs on `https://github.com/arsenal9971/MThesis/tree/master/Diagrams/results/EPIs`, the naming notation that we followed to identify them was **file_name**=$y_0$_$\epsilon$_$t_{max}$_$d_{max}$_$tp$_$f.png$ where $y_0$ is the $y$ position of the tracked points on the corresponding strip $\epsilon$ is half the width of the strip, $d_{max}$ is the used disparity for the corresponding sparse EPI, $tp$ is the number of tracked points in the strip and $f$ the number of features corresponding to the tracked points).

FIGURE 4.5: Strip of points of width $2\epsilon = 16.0$ px used for the benchmark to choose the disparity $d_{max}$

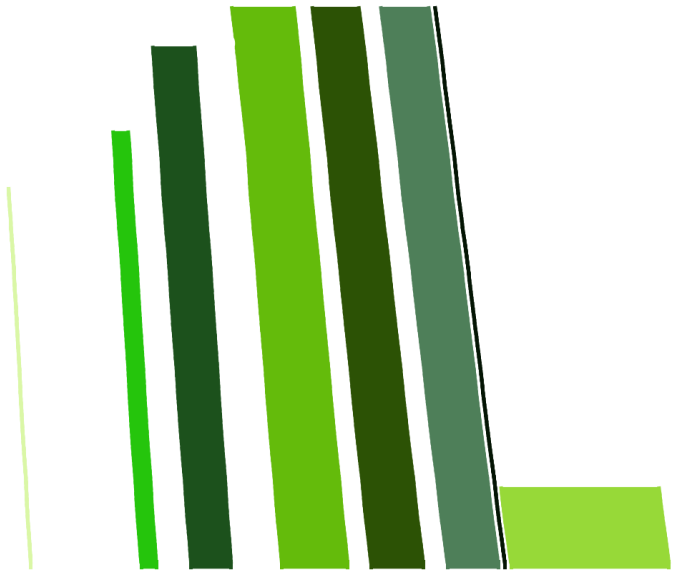


FIGURE 4.6: Densely sampled EPI associated to the points in the strip on Figure 4.5

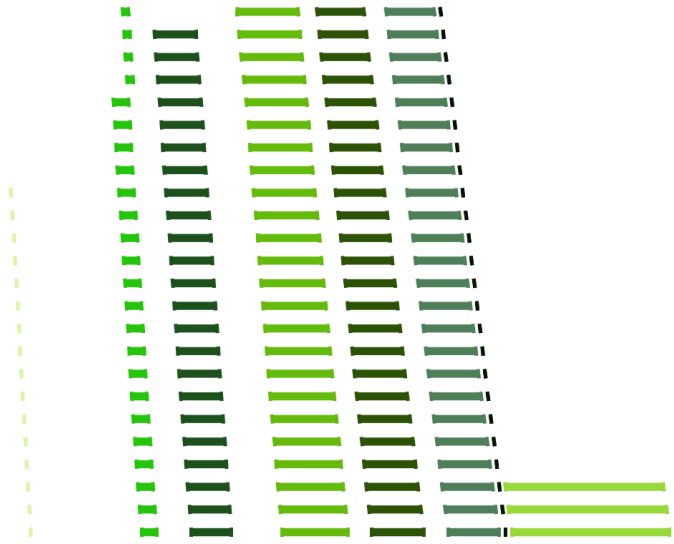


FIGURE 4.7: Sparsely sampled EPI associated to the disparity $d_{max} = 4$ px

One can also see in Figures 4.12, 4.13, 4.14 and 4.17 the lines detected in different inpainted EPIs corresponding to the same dense EPI, with different views disparity $d_{max}$, one can see that the important features are detected when $d_{max} = 4$ and $d_{max} = 7$ while when $d_{max}$ is bigger some lines are lost.

- **Third step (Inpaint sparse EPIs):** After paint all the sparsely sampled EPIs corresponding to different horizontal strips of width 16.0px whose union cover all the tracked points, we inpainted to recover a densely sampled EPIs and therefore we can finally compute the depth map by computing the slopes of the different lines in the EPIs. As we already mentioned on the last section, in order to perform the minimization problem associated with the inpainting algorithm we used iterative hard thresholding with a 0-Shearlet system as sparsifying system whose

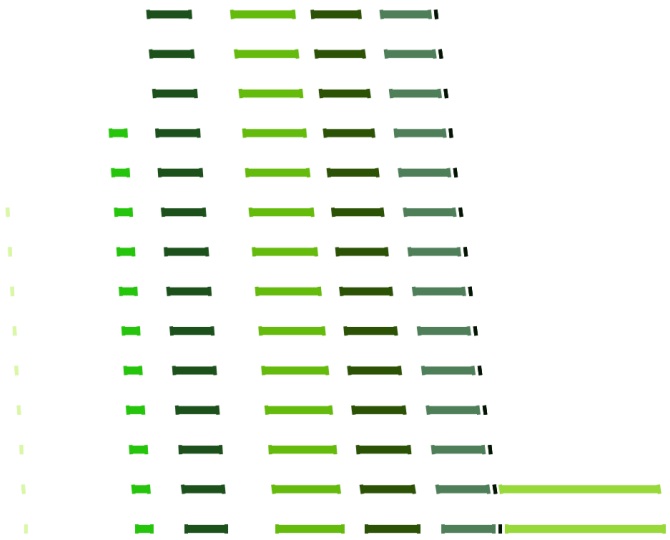


FIGURE 4.8: Sparsely sampled EPI associated to the disparity $d_{max} = 7$ px

FIGURE 4.9: Sparsely sampled EPI associated to the disparity $d_{max} = 9$ px

code we can find on Appendix C and was implemented using julia programming language with the libraries PyPlot.jl and Shearlab.jl.

We choose some of the best EPIs (corresponding to strips that capture an important number of tracked points) to show here (see Figures 4.18 to 4.37), but to compute the depth map we used all the inpainted EPIs that can be found on `https://github.com/arsenal9971/MThesis/tree/master/Diagrams/results/Inpainted` that follow the same naming notation as the dense EPIs, sparse EPIs and strips).

We are not looking here for high resolution light fields but for fast recoverable light fields so we imported in julia the sparse EPIs with a horizontal size of

FIGURE 4.10: Sparsely sampled EPI associated to the disparity $d_{max} = 12$ px

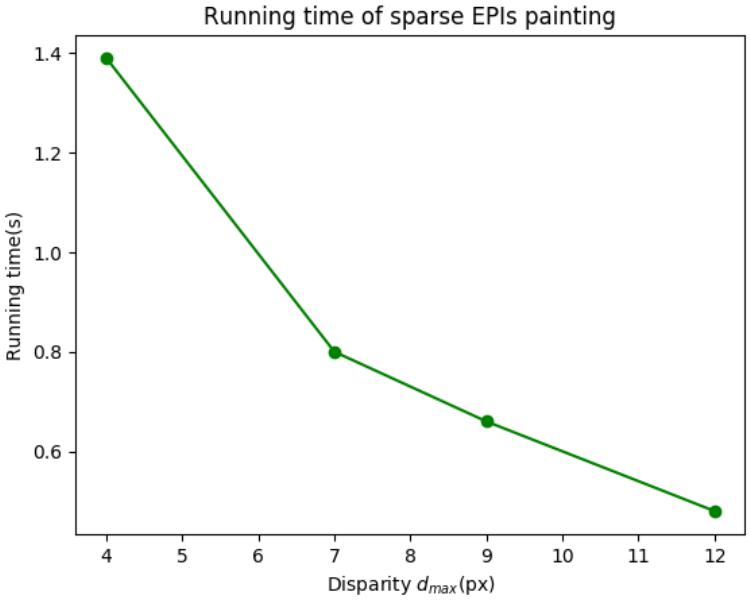



FIGURE 4.11: Running time of painting of sparsely sampled EPIs with different disparities

$n = 512$ pixels, using the same computer system as already mentioned the inpainting of the EPIs with 50 iterations took 19.15 seconds when using a fixed acceleration parameter $\alpha = 1$, whereas it took 15.5 seconds when using a the adaptive acceleration parameter defined on Algorithm 2.

- **Fourth step (line detection and depth map computation)**: We used Hough Line Transform on the inpainted EPIs to detect the lines and compute the depth map based on the slopes of the lines; this step will be explained in detail in the Section 4.3.

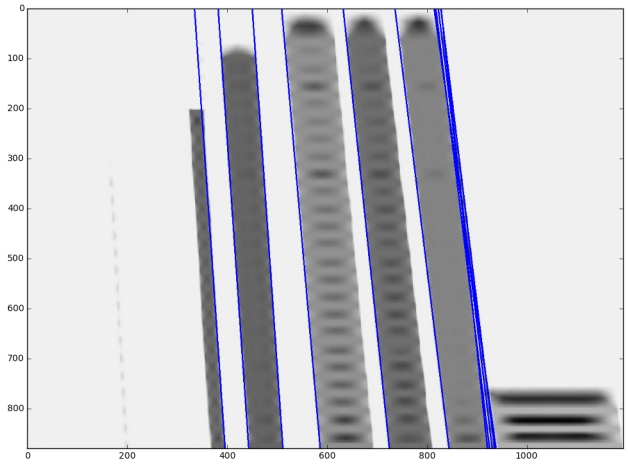



FIGURE 4.12: Lines detected in an inpainted using disparity $d_{max} = 4$ px

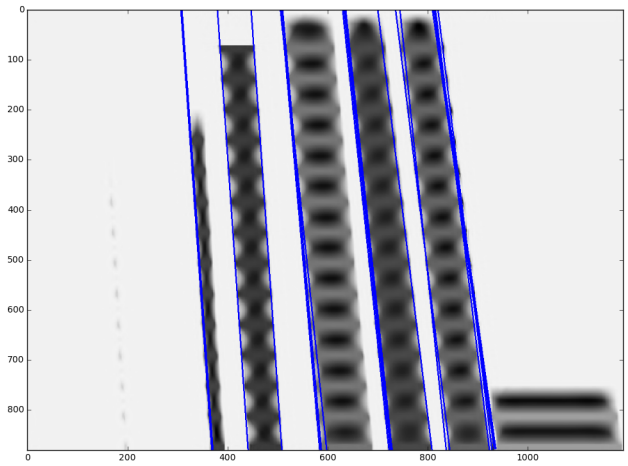



FIGURE 4.13: Lines detected in an inpainted using disparity $d_{max} = 7$ px

## 4.3 Line detection in inpainted EPIs and depth map computation

Once we have the inpaited EPIs for our sparse set of views with disparity $d_{max} = 7$ pixels, we need to be able to compute the slopes of the lines corresponding to different feature points, and then obtain the depth map; where the depth of a point corresponding to some line in an EPI will be given by Equation 2.2 in the form of

$$D = h\frac{\Delta X}{\Delta U} = h \cdot \text{slope} \tag{4.3}$$

where $h$ is the focal length of the camera that according to the specifications of the

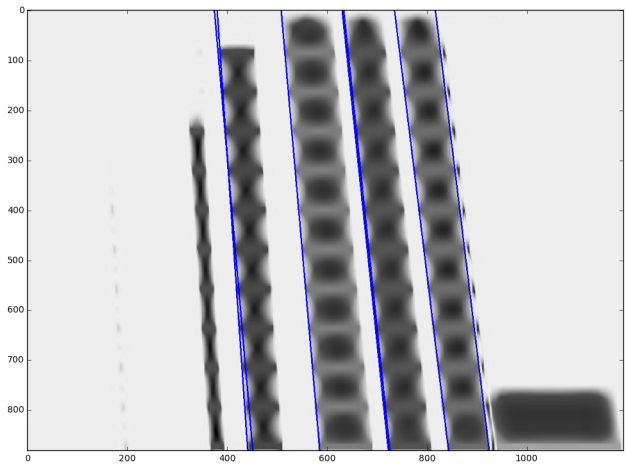



FIGURE 4.14: Lines detected in an inpainted using disparity $d_{max} = 9$ px

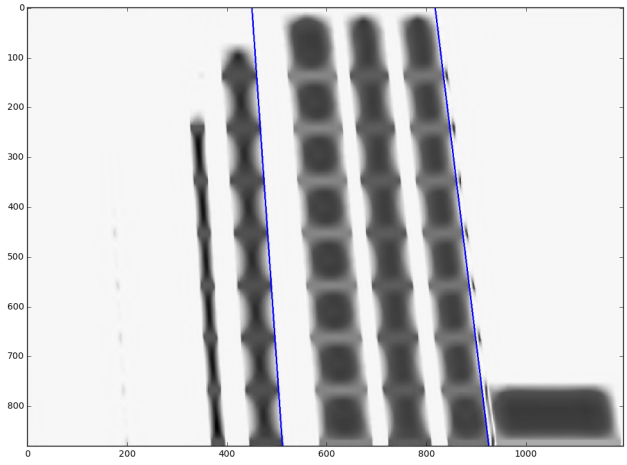



FIGURE 4.15: Lines detected in an inpainted using disparity $d_{max} = 12$ px

camera used in the data set is $h = 50mm = 5 \times 10^{-2}m$. In our case, each inpainted picture is imported in the line detection algorithm code as an image of certain $size = [size_x, size_y]$; by specifications of the data set we have that there were 101 taken pictures with a separation of 10mm from each other with a total time span of $1010mm = 1.01m$, the horizontal size in pixels of the used images in the data set is $683px$, therefore to obtain the depth of a point in Equation 4.3 in an unified way and be able to compare the obtained depths in different strips we will need to modify the equation with a normalization term multiplying $m$, obtaining the equation:

$$D = (0.01m) \cdot \text{slope} \cdot \left(\frac{\text{size}_x}{\text{size}_y}\right) \cdot \left(\frac{1.01m}{683px}\right) \tag{4.4}$$

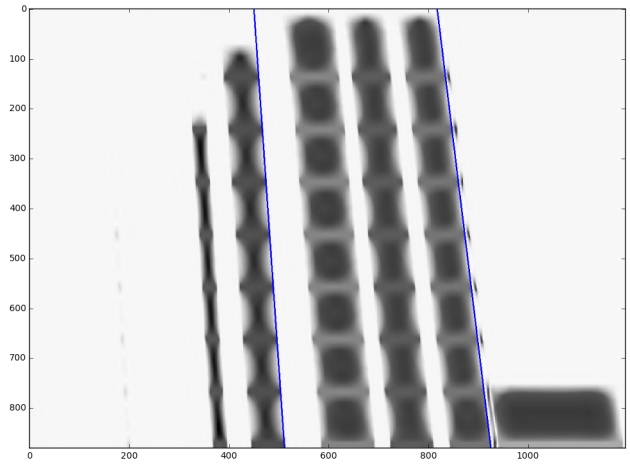



FIGURE 4.16: Lines detected in an inpainted using disparity $d_{max} = 12$ px

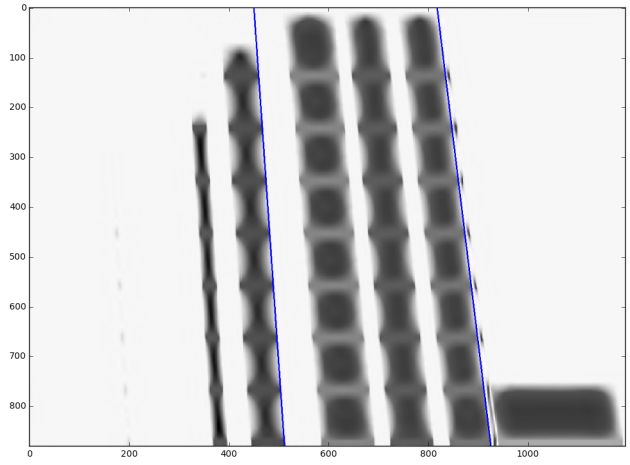



FIGURE 4.17: Lines detected in an inpainted using disparity $d_{max} = 12$ px

The resulting depth given by Equation 4.4 has units of $\frac{m^2}{px}$, the transformation from pixels to meters is not straight forward since we would need to know the transformation betweeen the spatial and image coordinates, and for that one should know a priori the actual depth of some fixed point and the correspondance in meters of the pixels of features at that depth; therefore the obtained depth is relative, but that is enough to construct the depth map.

Now that we know how to get the depth of points by its slope in the corresponding epipolar plane image we need to have a way to actually measure the slope of lines on the inpainted epipolar planes images and therefore we need a method to detect lines in an image. There are different methods to detect lines in images, commonly known as edge detectors; one can give as examples the Canny edge detector, the Phase Stretch

FIGURE 4.18: Strip with tracked points

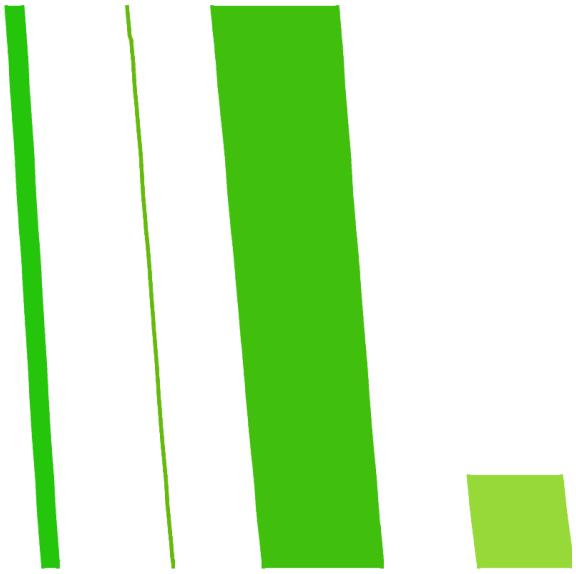


FIGURE 4.19: Dense EPI corresponding to the strip on Figure 4.18

Transform (PST), the Sobel method and the Hough Line Transform (see [28], [29], [64], [65] for a detailed explanation of each of this methods). By its easy and fast implementation and its effectiveness we used the Hough line transform to detect lines on the inpainted EPIs.

The Hough line transform was invented by Richard Duda and Peter Hart in 1972 on their paper "Use of the Hough Transformation to Detect Lines and Curves in Pictures"[64] and it is actually a generalization of a 1962 patent of Pual Hough[66] who proposed it for analysis of Bubble Chamber photographs for particle detection.

Even it seems as a trivial task for human vision, in automated analysis of digital images, a problem that often arises is the problem of detecting simple shapes as straight lines or circles. Generally an edge detector can be used as a pre-processing stage to obtain image points or image pixels that form part of the analysed curve in the image space. However imperfections in the image data or the edge detector may miss points or pixels on the desired cuves as well as spatial deviations between the ideal theoretical

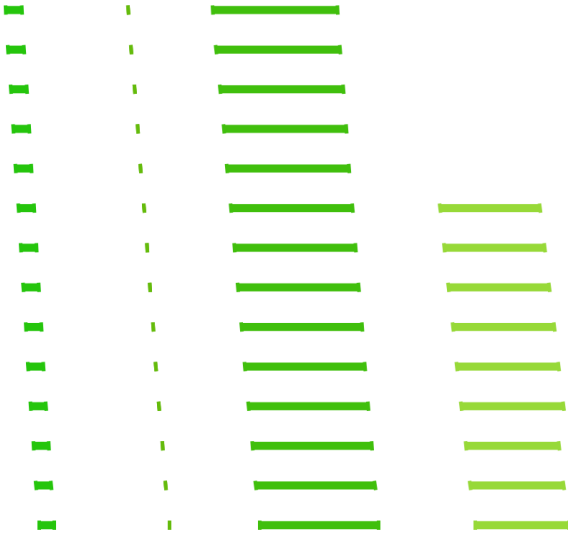


FIGURE 4.20: Sparse EPI corresponding to the strip on Figure 4.18

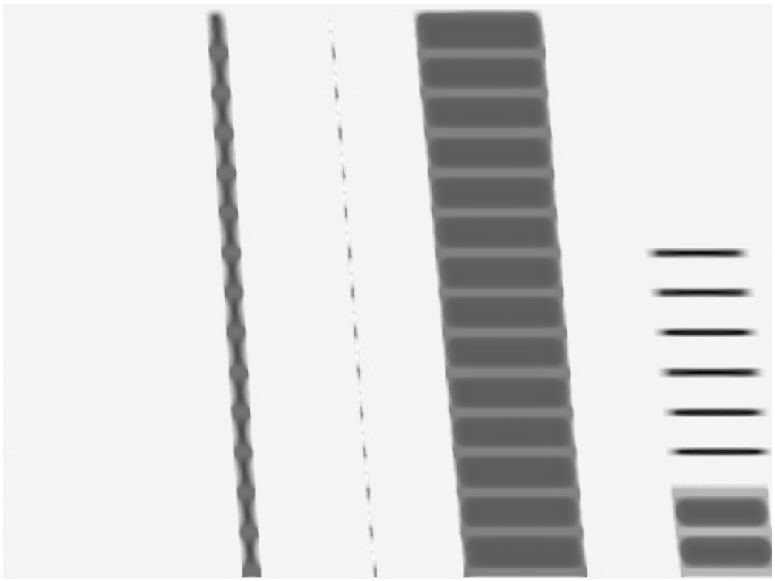


FIGURE 4.21: Inpainted EPI corresponding to the strip on Figure 4.18

line/circle and the noisy edge points as they are obtained from the edge detector; therefore it is non-trivial to group the extracted edge features to an appropriate set of lines or circles. The Hough transform has as purpose to address this problem by making it possible to perform groupings of edge points into object candidates by performing an explicit voting procedure over a set of parameterized image objects; in this thesis we will just make use of the simplest case of Hough transform that is detecting straight lines and it will be explained in the following.

A line in the image space can be expressed with two variables; for example by the parameters $(m, b)$ in Cartesian coordinates system where $m$ is the slope of the line and $b$ is the $y$-intercept of the line or in Polar coordinates system with parameters $(r, \theta)$ where $r$ is the distance to the origin and $\theta$ the angle with the $x$-axis. For Hough Transforms, we will express lines in the Polar system where a line equation can be written as:

FIGURE 4.22: Strip with tracked points

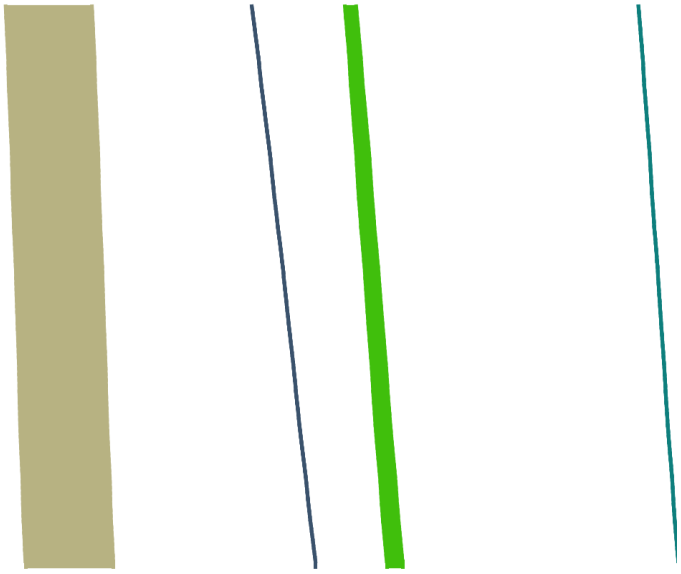


FIGURE 4.23: Dense EPI corresponding to the strip on Figure 4.22

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \tag{4.5}$$

this obtained by the relation $r = x\cos\theta + y\sin\theta$.

Given an arbitrary fixed point $(x_0, y_0) \in \mathbb{R}^2$, one can define a family of lines that goes through the points by

$$r_\theta = x_0\cos\theta + y_0\sin\theta \tag{4.6}$$

Therefore each pair $(r_\theta, \theta)$ represents each line that passes by $(x_0, y_0)$.

By the form of the Equation 4.6, for a given fixed point $(x_0, y_0)$ the plot of the family of lines that goes through it on the domain $\theta$ **vs.** $r_\theta$ is a sinusoid, we will consider only points such that $r > 0$ and $0 < \theta < 2\pi$(see Figure 4.38).

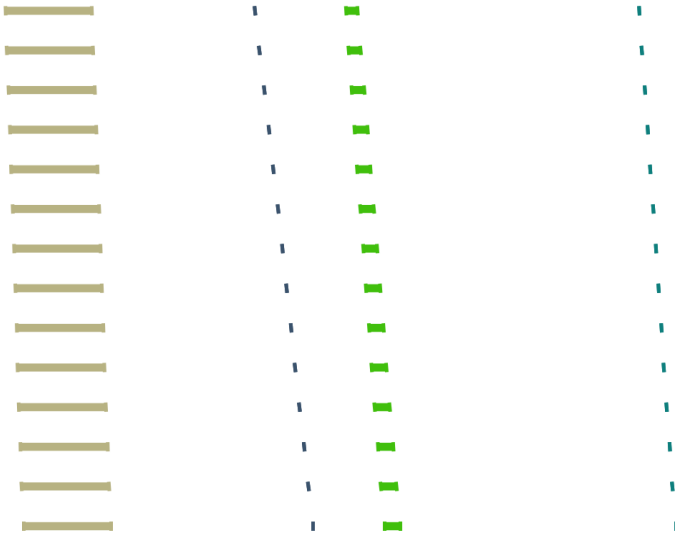


FIGURE 4.24: Sparse EPI corresponding to the strip on Figure 4.22

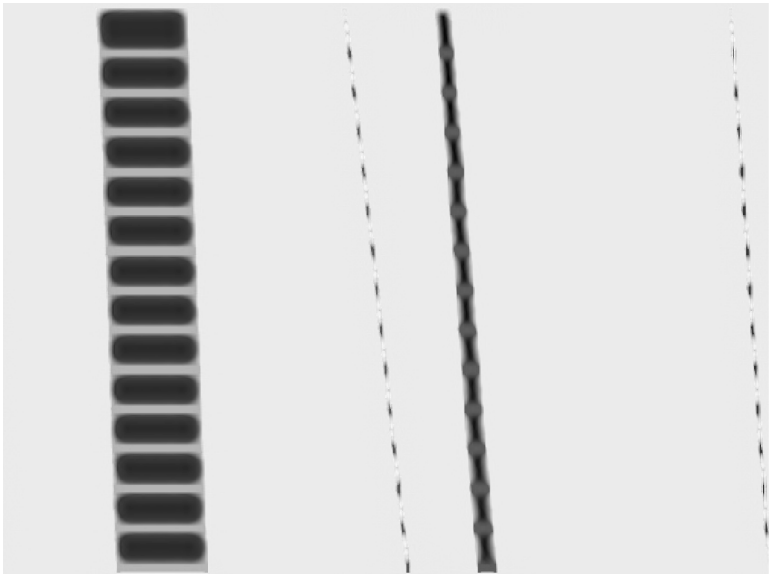


FIGURE 4.25: Inpainted EPI corresponding to the strip on Figure 4.22

We can plot the sinusoidal curves of all the points in an image. If the curves of two different points intersect in the plane $\theta - r_\theta$, it means that both points belong to a same line; see the example in Figure 4.39.

The three plots in Figure 4.39 intersect in one single point $(0.925, 9.6)$, these coordinates are the parameters $(\theta, r)$ or the line in which $(x_0, y_0)$, $(x_1, y_1)$ and $(x_2, y_2)$ lay; therefore in general, a line can be **detected** by finding the number of intersections between curves. The more curves intersecting means that the line represented by that intersection have more points. In general we can define a threshold of the minimum number of intersections needed to detect a line. This is what the Hough Line Transform does. It keeps track of the intersection between curves of every point in the image. If the number of intersections is above some threshold, then it declares it as a line with the parameters $(\theta, r_\theta)$ of the intersection point.

FIGURE 4.26: Strip with tracked points

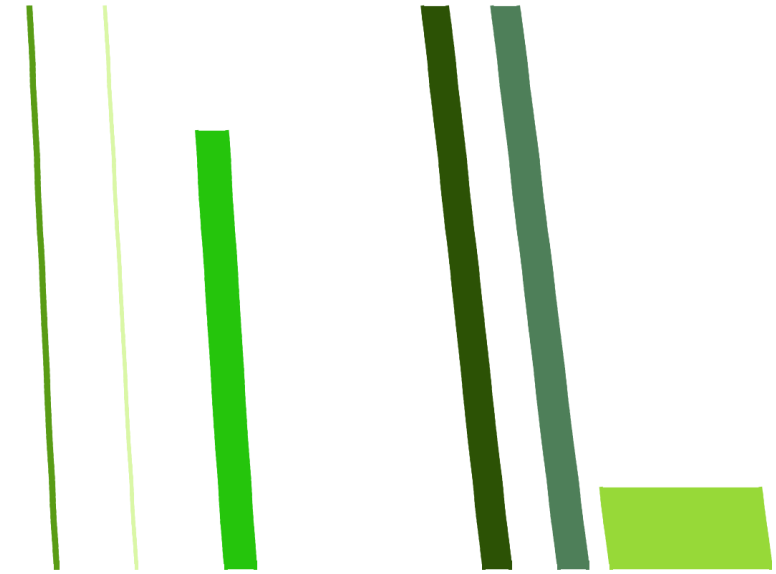


FIGURE 4.27: Dense EPI corresponding to the strip on Figure 4.26

There are different existing implementations of the Hough line, in Python, C++ and Matlab; in this thesis we used the python implementation using the OpenCV API, cause is very simple to use and fast to run; we made use of the Canny edge detector that is similar to the already explained Shi-Tomasi and for more detail one should check [28] and the Hough Line Transfom. In order to compute the depth map, we made use of the inpainted EPIs corresponding to strips at different fixed $y$'s; the points were divided by the different features were they belong, those features are enumerated in the following list and pictured in Figure 4.40

1. Bush 1.
2. Bush 2.
3. Bush 3.
4. Tree 1.

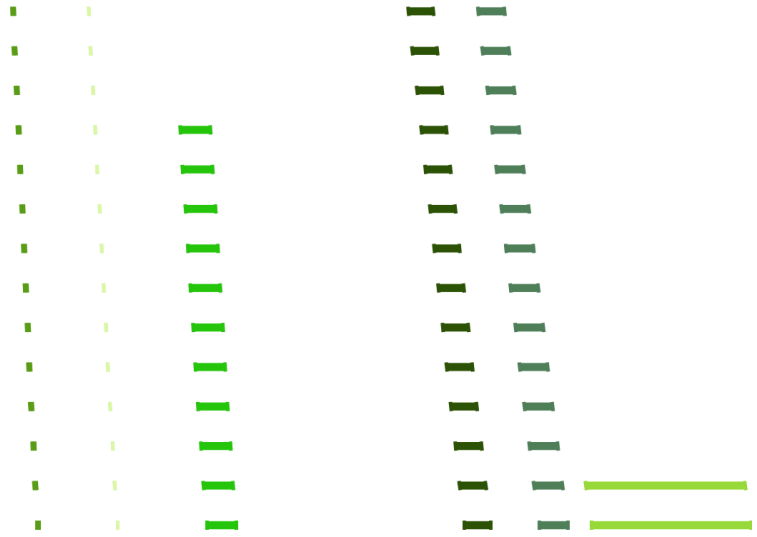


FIGURE 4.28: Sparse EPI corresponding to the strip on Figure 4.26

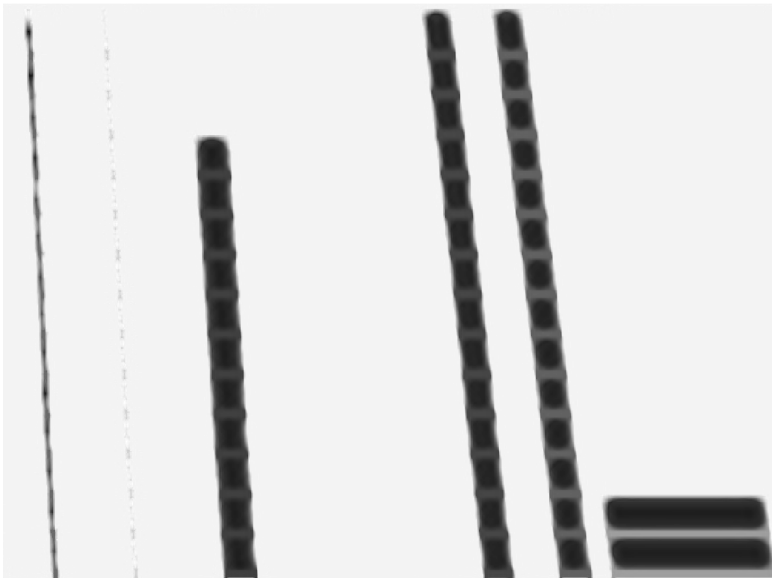


FIGURE 4.29: Inpainted EPI corresponding to the strip on Figure 4.26

5. Tree 2.
6. Tree 3.
7. Tree 4.
8. Tree 5.
9. Tree 6.
10. Lamp Post 1.
11. Lamp Post 2.
12. Front Wall.
13. Window 1.

FIGURE 4.30: Strip with tracked points

FIGURE 4.31: Dense EPI corresponding to the strip on Figure 4.30

14. Window 2.
15. Window 3.
16. Window 4.
17. Pillar 1.
18. Pillar 2.
19. Cable 1.
20. Cable 2.
21. Cable 3.
22. High Lamp.
23. Tower.

FIGURE 4.32: Sparse EPI corresponding to the strip on Figure 4.30

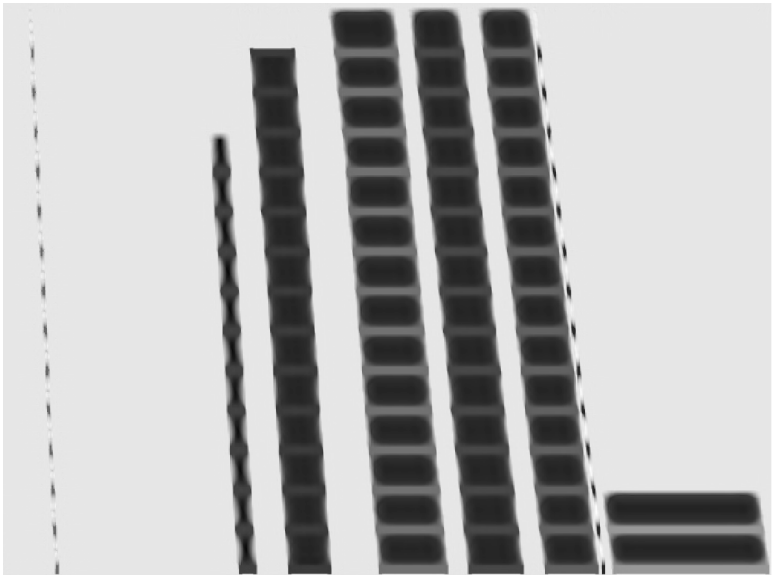


FIGURE 4.33: Inpainted EPI corresponding to the strip on Figure 4.30

Once we have points associated to features, we compute the depth of this points at different fixed $y$-strips finding out that for a fixed $y$-strip the points of a certain feature have relatively the same depth, therefore we assigned a depth to a feature for a fixed $y$. As we already mentioned we used the OpenCV python API as well as some python code for computing the depth map with this method, the code can be found in Appendix D and the obtained results are in the tables 4.1 to 4.6 where the values at each height $y$ (where the different strips are centered at) are the obtained depth of the points of that feature at that height in meters$^2$/pixels and whenever occur a blank space means that such feature has not points in such height (is worth to mention that the heights are measured from bottom to top, i.e. the bigger the height, the lower in the picture); this values describe completly the depth map, if one would like to visualize them in a refocusing-type application we should perform the so called light field rendering, where one finds contours of the corresponding features and apply a gaussian filter to defocused the elements that are not at the same depth, but this is

FIGURE 4.34: Strip with tracked points

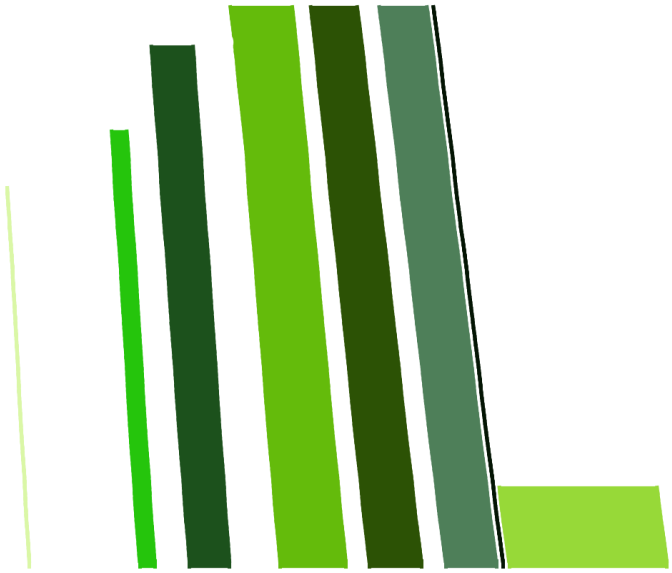


FIGURE 4.35: Dense EPI corresponding to the strip on Figure 4.34

not part of this thesis and will be taken as a future development goal. One can use as a guide of the heights Figure 4.41.

It could also be interesting to have depth map not by feature but by $x$ position, therefore one could make a heat map with the depth which will give us a more effective way to analyse it; the issue with this approach is that hardly more than the already tracked points can be tracked by the common tracking algorithms; this could be improved in the future using some other transformation like iterative feature extractors like Convolutional Neural Networks[67] to have a better corner detector and to be able to track more points; even though this approach works very well and the comparison with the existent light field recovery methods for approaching the depth map of a scene, in the next section we will shortly review this.

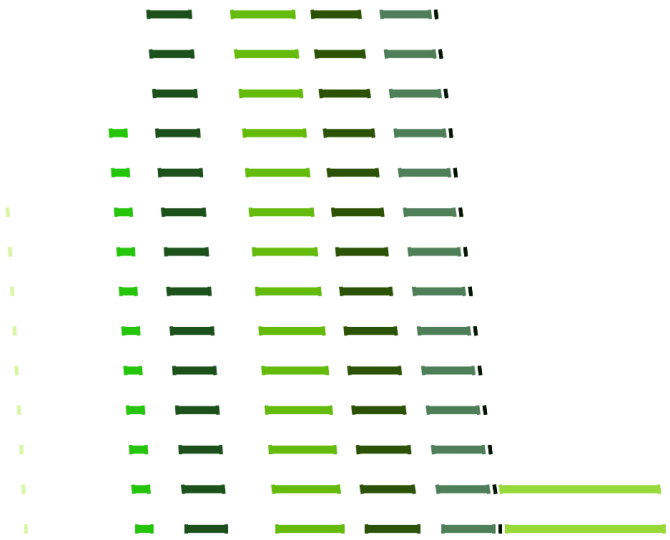


FIGURE 4.36: Sparse EPI corresponding to the strip on Figure 4.34

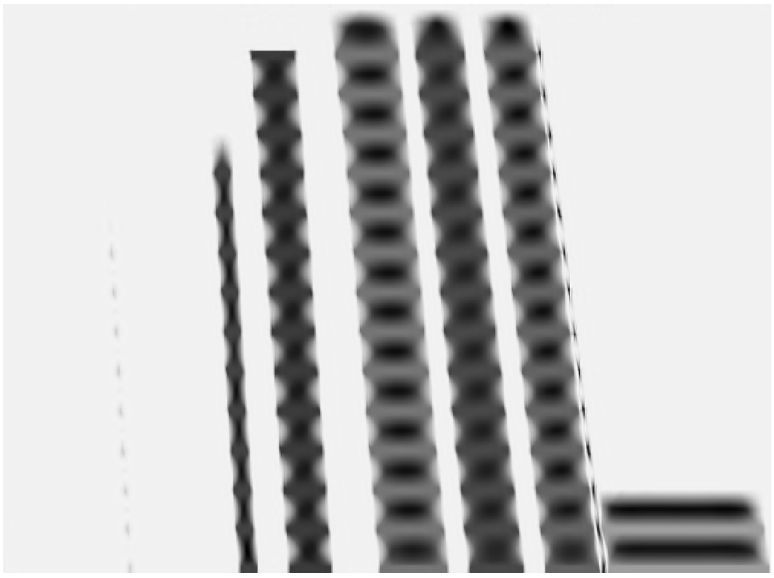


FIGURE 4.37: Inpainted EPI corresponding to the strip on Figure 4.34

## 4.4 Performance and results comparison with previews works

We already mentioned several times the computer system that was used is a low end MacBook whose processor and gpu are by far not the best in the commercial market and are even worse in comparison with the high-end servers used some researcher on the area.

In the last sections we got the next running times on finishing the whole pipeline of light field reconstruction:

- **Point tracking**: 16.36 seconds.
- **Sparse EPI painting**: 9.46 seconds (43 EPIs, 0.22 seconds for each).

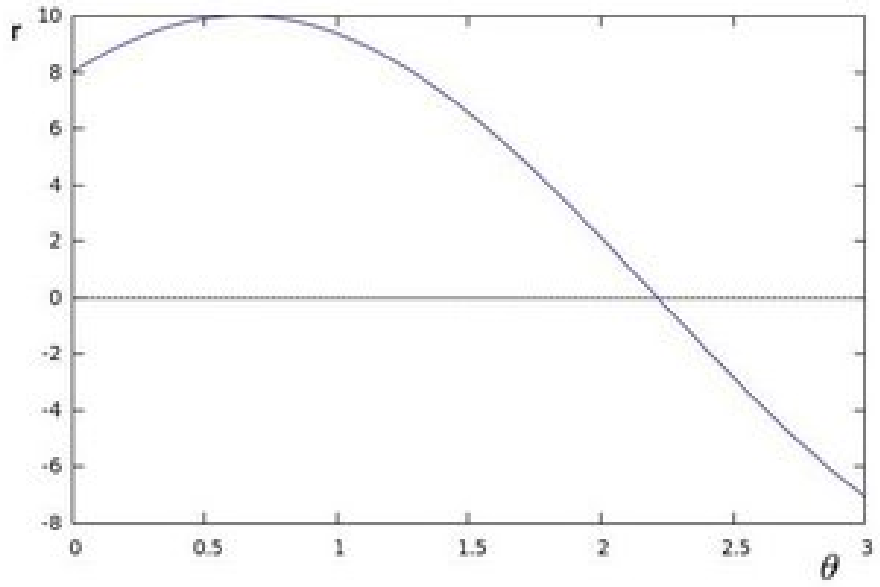



FIGURE 4.38: Plot of $\theta$vs.$r_\theta$ for $x_0 = 8$ and $y_0 = 6$

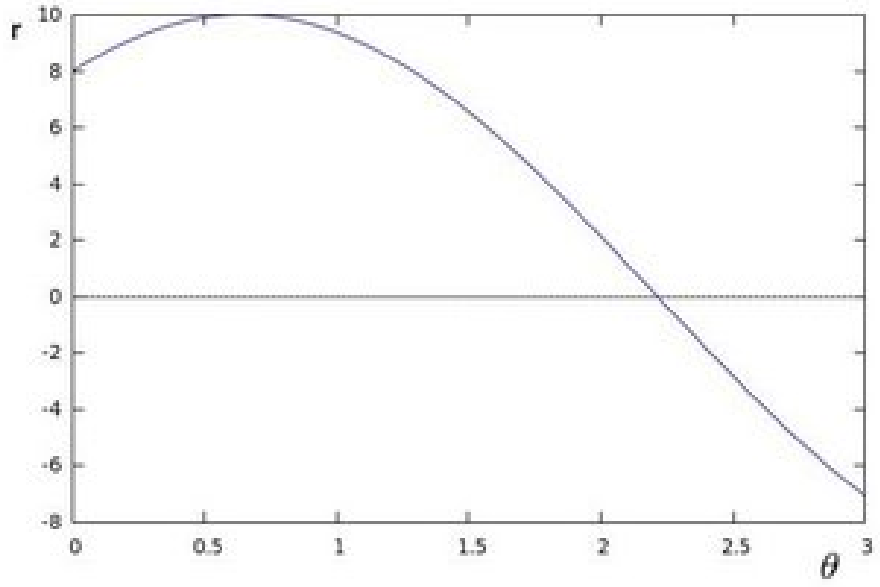



FIGURE 4.39: Plot of $\theta$vs.$r_\theta$ corresponding to the points $(x_0, y_0) = (8, 6)$, $(x_1, y_1) = (4, 9)$ and $(x_2, y_2) = (12, 3)$

- **Sparse EPI inpainting**: 666.50 (43 inpeinted EPIs, 15.50 seconds for each).
- **Line detection and depth map computation**: 0.43 seconds (43 EPIs, 0.01 seconds for each).

This gives a total of 692.75 seconds (around 11 seconds) to reconstruct the Light Field and compute the Depth Map. With this estimates we can make a comparison with the performance of the other works on light field reconstruction; to be clear it is important to mention that along the different approaches and works on light field reconstruction the goals of the methods are different although the main purpose is the same. In this particular thesis we wanted to present a method that is easy to implement, free, open source, fast and that doesn't require very sofisticated systems;

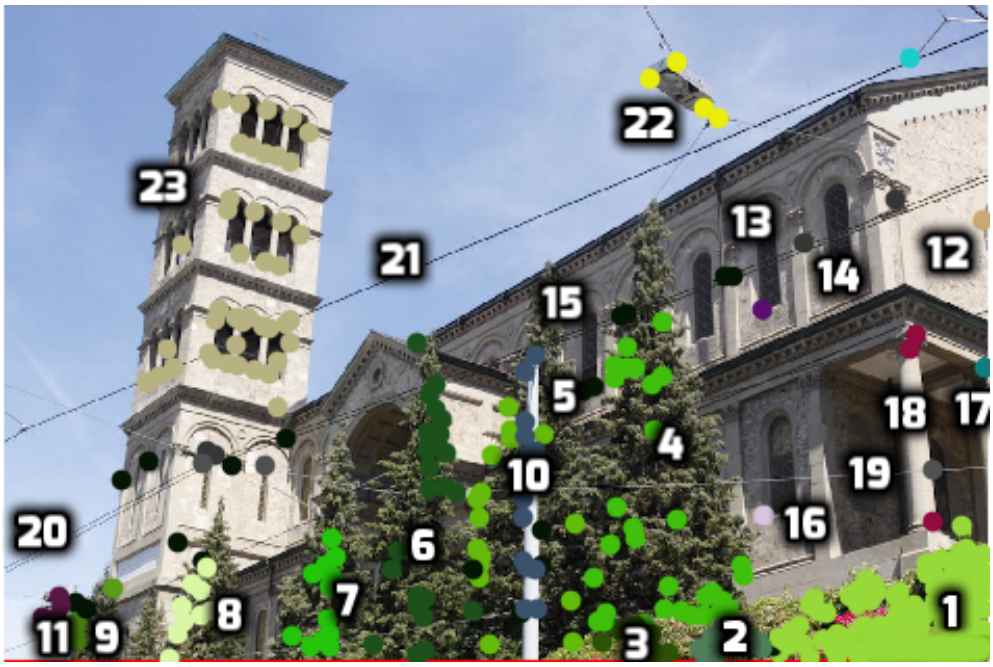



FIGURE 4.40: Features of the Church with tracked points

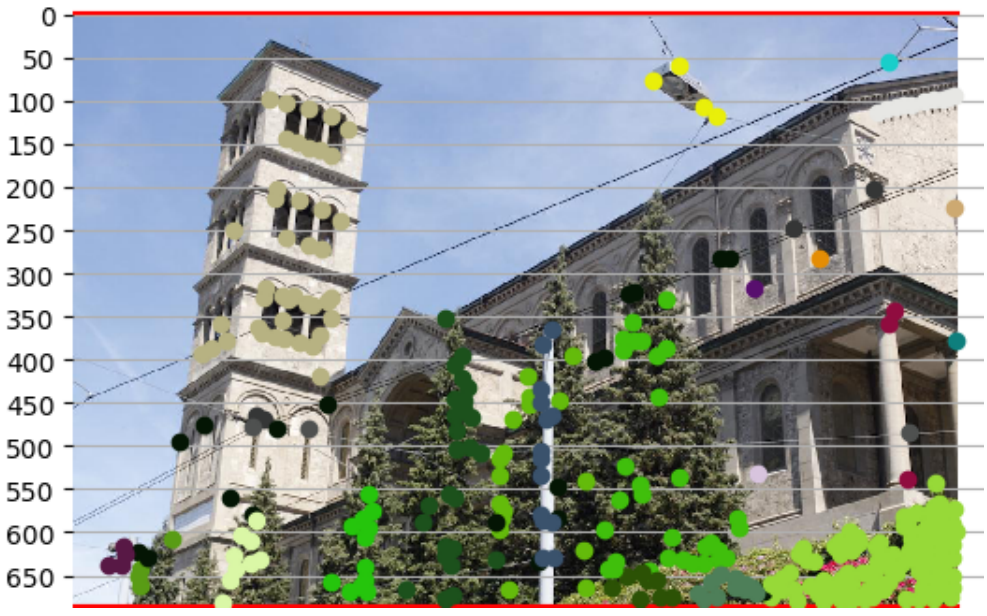



FIGURE 4.41: Features of the Church with tracked points with grid of heights

the limitations of our method comes when one wants to recover high definition light fields on images with few corners (few points to track with Lucas-Kanade), but for our particular dataset works perfectly. Having this in mind one can proceed with the comparison with the other works.

1. *"Compressive Light Field Photograph using Overcomplete Dictionaries and Optimized Projections"* (by Wetzstein et all, check [11]): Their approach made use of coded mask on an imaging lens to emulate a microlenses array on a SLR Camera (Nikon 105 mm f/2.8D). They learned a dictionary for 3D-light field with

| feature/height(px) | 680 | 673 | 664 | 632 | 616 | 600 | 584 | 568 |
|---|---|---|---|---|---|---|---|---|
| Bush1 | 0.0161 | 0.0157 | 0.0158 | 0.0156 | 0.0157 | 0.0161 | 0.0160 | 0.0158 |
| Bush2 | 0.0217 | 0.0221 | 0.0221 | 0.0220 | | | | |
| Bush3 | 0.0257 | 0.0261 | 0.0260 | 0.0258 | 0.0258 | | | |
| Tree1 | 0.0284 | 0.0285 | 0.0286 | 0.0285 | 0.0285 | 0.0286 | 0.0287 | 0.0291 |
| Tree2 | | | 0.0442 | | | 0.045 | 0.046 | 0.045 |
| Tree3 | 0.0501 | 0.0502 | 0.0501 | 0.0499 | 0.0498 | 0.0502 | 0.0500 | 0.0501 |
| Tree4 | 0.0594 | 0.0591 | 0.0590 | 0.0589 | 0.0590 | 0.0593 | 0.0595 | 0.0594 |
| Tree5 | 0.0801 | 0.0799 | 0.0798 | 0.0800 | 0.0801 | 0.0798 | 0.0799 | 0.0800 |
| Tree6 | 0.1001 | 0.0998 | 0.0999 | 0.0998 | 0.1003 | 0.1002 | | |
| LampPost1 | | | | 0.0181 | | 0.0180 | 0.0182 | |
| LampPost2 | | | | 0.1123 | 0.1122 | | | |
| FrontWall | | | | | | | | |
| Window1 | | | | | | | | |
| Window2 | | | | | | | | |
| Window3 | | | | | | | | |
| Window4 | | | | | | | | |
| Pillar1 | | | | | | | | |
| Pillar2 | | | | | | | | |
| Cable1 | | | | | | | | |
| Cable2 | | | | | | | | |
| Cable3 | | | | | | | | |
| HighLamp | | | | | | | | |
| Tower | | | | | | | | |

TABLE 4.1: Depth of features in $m^2/px$ at different heights in $px$

a data set of scenes (light field atoms) using an optimization problem as learning algorithm and then using this dictionary they performed compression of high resolution light fields using the LASSO algorithm [68] and the elements of the learned dictionary as representation system. In the implemenation they learned $1.7\times$ overcomplete dictionary consisting of $5,000$ light field atoms with a memory footprint of about 111 MB.

The employed the privative Sparse Modeling Software [69] on a workstation equipped with a 24-core Intel Xeon processor and 200 GB RAM, and it took about 10 hours. Finally the Light Field Sparse Reconstruction was performed in parallel on an 8-core Intel I7 workstation with 16 GB RAM, the reconstructions for three color channels took about 18 hours for each light field, which gave a total of 28 hours of processing for the reconstruction of one light field (almost 152 times slower than our method). Even this method is clearly slower than our approach the main difference is the resolution of the recovered light field almost 4 times higher than ours; though since it is based of a dictionary learned with a limited set of coded light fields, the recovery is very biased.

2. *"3D Reconstruction and Rendering from High Resolution Light Fields"* (ETH Zurich's PhD Thesis by C. Kim, check [6]): The data set that we used was actually acquired by the Research Group of Changil Kim on the ETH Zurich; the hardware was explained in detail on Section 2.5. They used the raw images with a resolution of $4080 \times 2720$ pixels, we used a downsampled version of the pictures of $1024 \times 683$ pixels. The geometric representation used by Kim is the

| feature/height(px) | 552 | 536 | 520 | 504 | 488 | 472 | 456 | 440 |
|---|---|---|---|---|---|---|---|---|
| Bush1 | 0.0157 | | | | | | | |
| Bush2 | | | | | | | | |
| Bush3 | | | | | | | | |
| Tree1 | 0.0291 | 0.0292 | 0.0290 | 0.0291 | | | | 0.030 |
| Tree2 | 0.0461 | 0.0456 | 0.0460 | 0.0459 | | | 0.0461 | 0.0461 |
| Tree3 | 0.0495 | 0.0496 | | 0.0497 | 0.0498 | 0.0499 | 0.0495 | 0.0501 |
| Tree4 | 0.0594 | | | | | | | |
| Tree5 | | | | | | | | |
| Tree6 | | | | | | | | |
| LampPost1 | 0.0181 | 0.0180 | 0.0182 | 0.0183 | | 0.0182 | 0.0179 | 0.0180 |
| LampPost2 | | | | | | | | |
| FrontWall | | | | | | | | |
| Window1 | | | | | | | | |
| Window2 | | | | | | | | |
| Window3 | | | | | | | | |
| Window4 | | 0.0391 | | | | | | |
| Pillar1 | | | | | | | | 0.0054 |
| Pillar2 | | 0.0251 | | | | | | |
| Cable1 | | | | | 0.0020 | | | |
| Cable2 | | | | | | 0.0701 | 0.0608 | |
| Cable3 | | | | | | | | |
| HighLamp | | | | | | | | |
| Tower | | | | | | | | |

TABLE 4.2: Depth of features in $m^2/px$ at different heights in $px$

same as we used, Epipolar Plane Images, but in this case densely sampled using a fine-to-coarse depth estimation (starting with parts of highest resolution and moving to less clear parts), this premits them to have a high resolution EPIs and have depth estimation in small neighbourhood of individual pixels. For the Light Field Recovery they used a desktop PC with an Intel Core i7 3.2 GHz and an NVidia GTX 680 graphics card, with a total of 100 views at 21 MP resolution the computation of a very detailed depth map took around 100 minutes (around 10 times slower than our method).

3. *"Light Field Reconstruction Using Shearlet Transform"* (by S. Vagharshakyan et al., check [3]): The recovery method and implementation on this thesis is strongly inspired on the work of Vagharshakyan; the biggest difference is that Vagharshakyan et al just explained the theory behind the Light Field reconstruction, but not the actual implementation of the whole pipeline, we cannot perform a proper benchmark of comparison with this work, since they just presented the PSNR (Peak Signal-to-Noise Ratio) as a performance reference; in our case it is not so important to have a high PSNR as long as the lines of the inpainted EPIs are clear enough to be detected with the Hough Line Transform, which actually happened.

   As in our case Vagharshakyan et al. used publicly available datasets since they did not have the necessary hardware to actually acquire the whole set of views. For the reconstruction algorithm they used two privative software: DERS (Depth Estimation Reference Software) and VSRS (View Synthesis Reference Software),

| feature/height(px) | 424 | 408 | 392 | 376 | 360 | 344 | 328 | 322 |
|---|---|---|---|---|---|---|---|---|
| Bush1 | | | | | | | | |
| Bush2 | | | | | | | | |
| Bush3 | | | | | | | | |
| Tree1 | | | 0.0332 | 0.0341 | 0.0341 | | 0.0335 | |
| Tree2 | | 0.0459 | | | | | | |
| Tree3 | 0.0502 | 0.0500 | 0.0501 | | 0.0502 | 0.0499 | | |
| Tree4 | | | | | | | | |
| Tree5 | | | | | | | | |
| Tree6 | | | | | | | | |
| LampPost1 | | | | | 0.0180 | 0.0181 | | |
| LampPost2 | | | | | | | | |
| FrontWall | | | | | | | | |
| Window1 | | | | | | | | 0.0601 |
| Window2 | | | | | | | | |
| Window3 | | 0.0522 | | | | | | |
| Window4 | | | | | | | | |
| Pillar1 | | | | | | | | |
| Pillar2 | | | | | | 0.0256 | | |
| Cable1 | | | | | | | | |
| Cable2 | | | | | | | 0.0212 | |
| Cable3 | | | | | | | | |
| HighLamp | | | | | | | | |
| Tower | 0.0995 | | 0.1002 | 0.1003 | 0.1002 | 0.1001 | 0.1002 | 0.0996 |

TABLE 4.3: Depth of features in $m^2/px$ at different heights in $px$

both developed as part of the Free Viewpoint TeleVision (FTV) project of the ISO/IEC Moving Pictures Experts Group MPEG (check [70] and [71] for a detailed explanation of this two pieces of software). We have no benchmarks of the reconstruction algorithm performed by VSRS+DERS but we know that Vagharshakyan et al. used as Shearlet System generator the matlab version of Shearlab 3D; in our case we used the julia version of Shearlab written by the autor of this thesis, it was already proven that the julia version is faster than the matlab version (check the benchmarks on `https://github.com/arsenal9971/Shearlab.jl/tree/master/benchmarks`), so it is expected that our implementation is faster than the one of Vagharshakyan and his group.

The last items just showed that the Light Field reconstruction method with the complete pipline presented in this thesis is speed-wise better than other methods proposed in the pass; even though our approach has a different goal than them at the end one would like to perform some features like image refocusing or depth estimation applied in computer vision (in particular autonomous driving) and our method is good enough for this purposes; of course we still can perform a lot of different applications with the estimated depth map, like light field 3D rendering and image refocusing but we will let this for future work; this and other planned future developments will be discussed in the next chapter.

| feature/height(px) | 312 | 306 | 300 | 292 | 280 | 264 | 248 | 232 |
|---|---|---|---|---|---|---|---|---|
| Bush1 | | | | | | | | |
| Bush2 | | | | | | | | |
| Bush3 | | | | | | | | |
| Tree1 | | | | | | | | |
| Tree2 | | | | | | | | |
| Tree3 | | | | | | | | |
| Tree4 | | | | | | | | |
| Tree5 | | | | | | | | |
| Tree6 | | | | | | | | |
| LampPost1 | | | | | | | | |
| LampPost2 | | | | | | | | |
| FrontWall | | | | | | | | 0.0051 |
| Window1 | | | | | | | | |
| Window2 | | | | | 0.0402 | | | |
| Window3 | | | | | | | | |
| Window4 | | | | | | | | |
| Pillar1 | | | | | | | | |
| Pillar2 | | | | | | | | |
| Cable1 | | | | | | | | |
| Cable2 | | | | | 0.0131 | | | |
| Cable3 | | | | | | | | |
| HighLamp | | | | | | | | |
| Tower | | | | | | 0.1001 | 0.0999 | 0.1003 |

TABLE 4.4: Depth of features in $m^2/px$ at different heights in $px$

| feature/height(px) | 216 | 200 | 168 | 152 | 136 | 120 | 104 | 88 |
|---|---|---|---|---|---|---|---|---|
| Bush1 | | | | | | | | |
| Bush2 | | | | | | | | |
| Bush3 | | | | | | | | |
| Tree1 | | | | | | | | |
| Tree2 | | | | | | | | |
| Tree3 | | | | | | | | |
| Tree4 | | | | | | | | |
| Tree5 | | | | | | | | |
| Tree6 | | | | | | | | |
| LampPost1 | | | | | | | | |
| LampPost2 | | | | | | | | |
| FrontWall | | | | | | | | |
| Window1 | | | | | | | | |
| Window2 | | | | | | | | |
| Window3 | | | | | | | | |
| Window4 | | | | | | | | |
| Pillar1 | | | | | | | | |
| Pillar2 | | | | | | | | |
| Cable1 | | | | | | | | |
| Cable2 | | 0.0015 | | | | | | |
| Cable3 | | | | | | | | |
| HighLamp | | | | | | 0.0009 | | |
| Tower | 0.1002 | 0.0998 | | 0.0999 | 0.1002 | 0.0998 | 0.1001 | |

TABLE 4.5: Depth of features in $m^2/px$ at different heights in $px$

| feature/height(px) | 72 | 56 |
|---|---|---|
| Bush1 | | |
| Bush2 | | |
| Bush3 | | |
| Tree1 | | |
| Tree2 | | |
| Tree3 | | |
| Tree4 | | |
| Tree5 | | |
| Tree6 | | |
| LampPost1 | | |
| LampPost2 | | |
| FrontWall | | |
| Window1 | | |
| Window2 | | |
| Window3 | | |
| Window4 | | |
| Pillar1 | | |
| Pillar2 | | |
| Cable1 | | |
| Cable2 | | |
| Cable3 | | 0.0020 |
| HighLamp | 0.0008 | 0.0007 |
| Tower | | |

TABLE 4.6: Depth of features in $m^2/px$ at different heights in $px$

# Chapter 5

# Conclusion and outlook

Template of Conclusion, [26]

# Appendices

# Appendix A

# Code for point tracking

The code used in the python API of OpenCV to detect the N strongest corners and track them through the 101 different views in the Church data set is presented in the following:

```
#! /usr/bin/python
# -*- coding: utf-8 -*-

# Import the libraries to use
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pandas as pd
from PIL import Image

# Path of the pictures with low resolution
path_lowres = './Church_data_set/church_image-raw/'
              +'church_image_lowres/'

# Parameters for Shi-Tomasi corner detection
feature_params = dict( maxCorners = 400, # A max. of 400 strong
                                         # corners
                       qualityLevel = 0.3,
                       minDistance = 7,
                       blockSize = 7 )
# Parameters for Lucas-Kanade optical flow
lk_params = dict( winSize  = (18,18),
                  maxLevel = 2,
                  criteria = (cv2.TERM_CRITERIA_EPS
                              | cv2.TERM_CRITERIA_COUNT
                              , 10, 0.03))
# Create some random colors
color = np.random.randint(0,255,(100,3))

# Take first frame image and find corners in church data set
old_frame = cv2.imread(path_lowres+'church_image-raw_0000'
                                  +'_lowres.jpg',1)
old_gray = cv2.cvtColor(old_frame, cv2.COLOR_BGR2GRAY)
p0 = cv2.goodFeaturesToTrack(old_gray, mask = None,
                             **feature_params)
st = np.array([[1]]*len(p0))
# Create a data frame with the entries of the point
```

```
df_church = pd.DataFrame({'x1' : p0[st==1][:,0],
                          'y1' : p0[st==1][:,1]})
# Vector to track the number of points to track
lengths_church = [len(p0)]
# Create a mask image for drawing purposes
mask = np.zeros_like(old_frame)

# For loop to track those points with Lucas-Kanade
for i in range(0,100):
    if i < 10:
        frame = cv2.imread(path_lowres+'church_image-raw_000'
                           +str(i)+'_lowres.jpg')
        frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # calculate optical flow
        p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray,
                      frame_gray, p0, None, **lk_params)
        # Select good points
        good_new = p1[st==1]
        good_old = p0[st==1]
        # append new length to lengths
        lengths_church.append(len(p1))

        # Adding the new points to the dataframe
        df_church_new = pd.DataFrame({
                    'x'+str(i+2) : [np.nan]*len(df_church.x1),
                     'y'+str(i+2) : [np.nan]*len(df_church.x1)})
        notnull = ~pd.isnull(df_church['x'+str(i+1)])
        df_church_new1 = df_church_new[notnull]
        id=df_church_new1[[sti[0]==1 for sti in st]].index
        df_church_new1['x'+str(i+2)][[sti[0]==1 for sti in st]]
                        = pd.Series(p1[st==1][:,0],index = id)
        df_church_new1['y'+str(i+2)][[sti[0]==1 for sti in st]]
                         = pd.Series(p1[st==1][:,1],index = id)
        df_church_new[notnull] = df_church_new1
        df_church=df_church.join(df_church_new)

        # draw the tracks
        for j,(new,old) in enumerate(zip(good_new,good_old)):
            a,b = new.ravel()
            c,d = old.ravel()
            mask = cv2.line(mask, (a,b),(c,d), color[j%100].tolist()
                                  , 2)
            frame = cv2.circle(frame,(a,b),5,color[j%100].tolist()
                                  , -1)
        img = cv2.add(frame,mask)
        #cv2.imshow('frame',img)

        # Now update the previous frame and previous points
        old_gray = frame_gray.copy()
        p0 = good_new.reshape(-1,1,2)
```

```
    else:
        if i < 100:
            frame = cv2.imread(path_lowres
                    +'church_image-raw_00'+str(i)+'_lowres.jpg')
            frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            # calculate optical flow
            p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray,
                          p0, None, **lk_params)
            # Select good points
            good_new = p1[st==1]
            good_old = p0[st==1]
            # append new length to lengths
            lengths_church.append(len(p1))

            # Adding the new points to the dataframe
            df_church_new = pd.DataFrame({'x'+str(i+2) : [np.nan]
                *len(df_church.x1), 'y'+str(i+2) :
                 [np.nan]*len(df_church.x1)})
            notnull = ~pd.isnull(df_church['x'+str(i+1)])
            df_church_new1 = df_church_new[notnull]
            id=df_church_new1[[sti[0]==1 for sti in st]].index
            df_church_new1['x'+str(i+2)][[sti[0]==1 for sti in st]]
             =  pd.Series(p1[st==1][:,0],index = id)
            df_church_new1['y'+str(i+2)][[sti[0]==1 for sti in st]]
             = pd.Series(p1[st==1][:,1],index = id)
            df_church_new[notnull] = df_church_new1
            df_church=df_church.join(df_church_new)

            # draw the tracks
            for j,(new,old) in enumerate(zip(good_new,good_old)):
                a,b = new.ravel()
                c,d = old.ravel()
                mask = cv2.line(mask, (a,b),(c,d),
                                color[j%100].tolist(), 2)
                frame = cv2.circle(frame,(a,b),5,
                                color[j%100].tolist(),-1)
            img = cv2.add(frame,mask)
            #cv2.imshow('frame',img)

            # Now update the previous frame and previous points
            old_gray = frame_gray.copy()
            p0 = good_new.reshape(-1,1,2)
        else:
            frame = cv2.imread(path_lowres+'church_image-raw_0100_'
                                          +'lowres.jpg')
            frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            # calculate optical flow
            p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray,
                          p0, None, **lk_params)
            # Select good points
            good_new = p1[st==1]
```

```
            good_old = p0[st==1]
            # append new length to lengths
            lengths_church.append(len(p1))

            # Adding the new points to the dataframe
            df_church_new = pd.DataFrame({'x'+str(i+2) : [np.nan]
              *len(df_church.x1), 'y'+str(i+2) : [np.nan]*len(df_church.x1)})
            notnull = ~pd.isnull(df_church['x'+str(i+1)])
            df_church_new1 = df_church_new[notnull]
            id=df_church_new1[[sti[0]==1 for sti in st]].index
            df_church_new1['x'+str(i+2)][[sti[0]==1 for sti in st]]
            =  pd.Series(p1[st==1][:,0],index = id)
            df_church_new1['y'+str(i+2)][[sti[0]==1 for sti in st]]
            = pd.Series(p1[st==1][:,1],index = id)
            df_church_new[notnull] = df_church_new1
            df_church=df_church.join(df_church_new)

            # draw the tracks
            for j,(new,old) in enumerate(zip(good_new,good_old)):
                a,b = new.ravel()
                c,d = old.ravel()
                mask = cv2.line(mask, (a,b),(c,d),
                                color[j%100].tolist(), 2)
                frame = cv2.circle(frame,(a,b),5,
                                   color[j%100].tolist(),-1)
            img = cv2.add(frame,mask)
            img = cv2.add(frame,mask)
            #cv2.imshow('frame',img)

            # Now update the previous frame and previous points
            old_gray = frame_gray.copy()
            p0 = good_new.reshape(-1,1,2)

cv2.destroyAllWindows()

# Save in a csv file
df_church.to_csv('church_tracking.csv')

# Plot it
plt.rcParams["figure.figsize"] = [12,9]
plt.imshow(img,cmap='gray')
plt.show()
```

# Appendix B

# Code for painting EPIs

The following code was used to paint the dense and sparse EPIs corresponding to different fixed $y$ tracked points, with horizontal bands with width $2\epsilon = 12.0$.

```
#! /usr/bin/julia

# We will use library DataFrames to deal with DataFrames in csv
using DataFrames
using PyPlot

# Reading the table and images
points_x = readtable("/Users/hector/Documents/Master_thesis/
EPI_code/church_EPIs_x.csv")
points_y = readtable("/Users/hector/Documents/Master_thesis/
EPI_code/church_EPIs_y.csv");
# Import the first image
church_first = imread("/Users/hector/Documents/Master_thesis/
EPI_samples/Church_data_set/church_image-raw/
church_image_lowres/church_image-raw_0000_lowres.jpg")
church_last = imread("/Users/hector/Documents/Master_thesis/
EPI_samples/Church_data_set/church_image-raw/
church_image_lowres/church_image-raw_0100_lowres.jpg");

# Function that paints EPI
function paint_epi(points_x, points_y, epsilon, y0, s_rate=1)
    # Limits of the strip
    y01 = y0-epsilon;
    y02 = y0+epsilon;
    #Subset of features catched
    idy = (points_y[:y1].<=y02).*(points_y[:y1].>=y01);
    array_x = points_x[idy,:]
    array_y = points_y[idy,:];
    features = unique(array_x[:feature])
    for feature in unique(array_x[:feature])
        array_feature_x = array_x[array_x[:feature].
==feature,:];
        array_feature_y = array_y[array_y[:feature].
==feature,:];
        color_feature = array_x[array_x[:feature].==feature,:
][:color][1]

        # find min and max of x coordinate for that feature
```

```
        xmax_feature_first,idxmax_feature =
findmax(array_feature_x[:x1])
        xmin_feature_first, idxmin_feature =
findmin(array_feature_x[:x1]);

        nonas = min(array_feature_x[idxmax_feature,:][:

no_nas][1],array_feature_x[idxmin_feature,:][:no_nas][1]);
        i = 1
        max_size = size(array_feature_x[idxmax_feature,:])[2]
        cond_max = abs(array_feature_x[idxmax_feature,:][i+
s_rate]-array_feature_x[idxmax_feature,:][i+s_rate+1])[1]<=10.
0
        cond_min = abs(array_feature_x[idxmin_feature,:][i+
s_rate]-array_feature_x[idxmin_feature,:][i+s_rate+
1])[1]<=10.
0
        while (i <= (nonas-s_rate-2))*(cond_max)*(cond_min)
            x_max = [array_feature_x[idxmax_feature,:][i],
array_feature_x[idxmax_feature,:][i+1]]
            x_min = [array_feature_x[idxmin_feature,:][i],
array_feature_x[idxmin_feature,:][i+1]]
            plot(x_max,[i,i+1],color=color_feature)
            plot(x_min,[i,i+1],color=color_feature)
            x = [x_max[2][1],x_max[1][1],x_min[1][1],
x_min[2][1]]
            y = [i+1,i,i,i+1];
            fill(x,y,color=color_feature)
            x_max = [array_feature_x[idxmax_feature,:][i],
array_feature_x[idxmax_feature,:][i+1]]
            x_min = [array_feature_x[idxmin_feature,:][i],
array_feature_x[idxmin_feature,:][i+1]]
            cond_max = abs(array_feature_x[idxmax_feature,:
][i+s_rate]-array_feature_x[idxmax_feature,:][i+s_rate+1])[1]<=10.0

            cond_min = abs(array_feature_x[idxmin_feature,:
][i+s_rate]-array_feature_x[idxmin_feature,:][i+s_rate+
1])[1]<=10.0
            i+=s_rate
        end
    end
    plot([0,1024,1024,0,0],[0,0,maximum(array_x[:no_nas]),
maximum(array_x[:no_nas]),0],color="black")
    ax = gca()
    ax[:set_frame_on](false)
    ax[:set_frame_on](false)
    yticks([])
    xticks([])
    maximum(array_x[:no_nas])
end
```

```
# Function that paints a strip in the first image with the
# captured points
function strip_epi(points_x, points_y, epsilon, y0)
    # Limits of the strip
    y01 = y0-epsilon;
    y02 = y0+epsilon;
    #Subset of features catched
    idy = (points_y[:y1].<=y02).*(points_y[:y1].>=y01);
    array_x = points_x[idy,:]
    array_y = points_y[idy,:];
    plot([0,1024],[y01,y01],"-r")
    plot([0,1024],[y02,y02],"-r")
    i = 1
    imshow(church_first)
    x = array_x[i][~isna(array_x[i])]
    y = array_y[i][~isna(array_y[i])]
    colors = array_x[:color][~isna(array_x[i])]
    for j in 1:size(x)[1]
        plot(x[j],y[j], color=colors[j],"o")
    end
    ax = gca()
    ax[:set_frame_on](false)
    ax[:set_frame_on](false)
    yticks([])
    xticks([])
end

feature = "bush1"
y0 = maximum(points_y[points_y[:feature].==feature,:][:y1])
# Half the width of the strip
epsilon = 8.0;
# Disparity
s_rate = 7;

## Lets paint the corresponding EPI
for y00 in y0:-2epsilon:0+epsilon
    # Limits of the strip
    y01 = y00-epsilon;
    y02 = y00+epsilon;
    #Subset of features catched
    idy = (points_y[:y1].<=y02).*(points_y[:y1].>=y01);
    array_x = points_x[idy,:];
    t_max = maximum(array_x[:no_nas])
    no_features = size(unique(array_x[:feature]))[1]
    no_features = size(unique(array_x[:feature]))[1]
    tracked_points = size(array_x)[1];
    name=string(Int(y00))*"_"*string(Int(epsilon))*"_"*
string(t_max)*"_"string(s_rate)*"_"*string(tracked_points)*
"_"*string(no_features)
    path="/Users/hector/Documents/Github_Repos/MThesis/
Diagrams/results/EPIs/"
```

```
    strip_epi(points_x, points_y,epsilon,y00)
    savefig(path*name*"_strip", dpi = 80*3,
bbox_inches="tight")
    clf()
    paint_epi(points_x, points_y, epsilon, y00,s_rate)
    savefig(path*name*"_sparse", dpi = 80*3,
bbox_inches="tight")
    clf()
    paint_epi(points_x, points_y, epsilon, y00,1)
    savefig(path*name*"_dense", dpi = 80*3,
bbox_inches="tight")
    clf()
end
```

# Appendix C

# Code for inpainting sparse EPIs

The following code was used to inpaint the sparse EPIs with iterative hard thresholding as minimization algorithm and 0-Shearlets as sparsifying syste; it was implemented using Julia programming language with the libraries PyPlot.jl and Shearlab.jl

```
#! /usr/bin/julia

# Calling the libraries
using PyPlot
using Shearlab

# We will import the sparse EPI with 512 horizontal pixels
n =512;

# Function to inpaint EPIs using hard normalized iterative
# thersholding with constant acceleration parameter \alpha=1
function inpaint2D(imgMasked,mask,iterations,stopFactor,
shearletsystem)
    coeffs = Shearlab.sheardec2D(imgMasked,shearletsystem);
    coeffsNormalized = zeros(size(coeffs))+im*
zeros(size(coeffs));
    for i in 1:shearletsystem.nShearlets
        coeffsNormalized[:,:,i] = coeffs[:,:,i]./
shearletsystem.RMS[i];
    end
    delta = maximum(abs(coeffsNormalized[:]));
    lambda=(stopFactor)^(1/(iterations-1));
    imgInpainted = zeros(size(imgMasked));
    #iterative thresholding
    for it = 1:iterations
        res = mask.*(imgMasked-imgInpainted);
        coeffs = Shearlab.sheardec2D(imgInpainted+res,
shearletsystem);
        coeffsNormalized = zeros(size(coeffs))+im*
zeros(size(coeffs));
        for i in 1:shearletsystem.nShearlets
            coeffsNormalized[:,:,i] = coeffs[:,:,i]./
shearletsystem.RMS[i];
        end
        coeffs = coeffs.*(abs(coeffsNormalized).>delta);
        imgInpainted = Shearlab.shearrec2D(coeffs,
shearletsystem);
```

```
        delta=delta*lambda;
    end
    imgInpainted
end

# Function to inpaint EPIs using hard normalized iterative
# thersholding with adaptive acceleration parameter
function inpaint2D_accel(imgMasked,mask,iterations,stopFactor,
shearletsystem)
    coeffs = Shearlab.sheardec2D(imgMasked,shearletsystem);
    coeffsNormalized = zeros(size(coeffs))+im*
zeros(size(coeffs));
    for i in 1:shearletsystem.nShearlets
        coeffsNormalized[:,:,i] = coeffs[:,:,i]./
shearletsystem.RMS[i];
    end
    delta = maximum(abs(coeffsNormalized[:]));
    lambda=(stopFactor)^(1/(iterations-1));
    imgInpainted = zeros(size(imgMasked));
    alpha = 1
    #iterative thresholding
    for it = 1:iterations
        res = mask.*(imgMasked-imgInpainted);
        coeffs = Shearlab.sheardec2D(imgInpainted+alpha*res,
shearletsystem);
        coeffsNormalized = zeros(size(coeffs))+im*
zeros(size(coeffs));
        for i in 1:shearletsystem.nShearlets
            coeffsNormalized[:,:,i] = coeffs[:,:,i]./
shearletsystem.RMS[i];
        end
        coeffs = coeffs.*(abs(coeffsNormalized).>delta);
        # Support of coeffsNormalized until
        norms = [norm(coeffs[:,:,i]) for i in 1:
size(coeffs)[3]]
        # Thresholding the norm of the matrices to catch the
#support (smallest )
        indices = (norms.<0.5)
        beta = Shearlab.sheardec2D(res,shearletsystem);
        beta[(~indices)...] = 0;
        beta2 = mask.*(Shearlab.shearrec2D(beta,
shearletsystem));
        alpha = sum((abs.(beta)).^2)/sum((abs.(beta2)).^2)
        imgInpainted = Shearlab.shearrec2D(coeffs,
shearletsystem);
        delta=delta*lambda;
    end
    imgInpainted
end

# List of sparse and dense EPIs
```

```
split_list(string)=split(string,"_")
list_files = readdir("../Diagrams/results/EPIs/");
list_files = list_files[2:length(list_files)]
sparse_list = [];
dense_list = [];

for string in list_files
    if split(string,"_")[7]=="sparse.png"
        push!(sparse_list,string)
    end
end

for string in list_files
    if split(string,"_")[7]=="dense.png"
        push!(dense_list,string)
    end
end


# Function to inpaint a specific file of sparse EPI with
# iterative thresholding with 50 iterations
inpaint_file(file_sparse,file_dense)
    # The path of the sparse EPI
    name_sparse = "../Diagrams/results/EPIs/"*file_sparse;
    EPI_sparse = Shearlab.load_image(name_sparse, n,m);
    EPI_sparse = EPI_sparse[:,:,1];
    name_dense = "../Diagrams/results/EPIs/"*file_dense;
    EPI_dense = Shearlab.load_image(name_dense, n,m);
    EPI_dense = EPI_dense[:,:,1];
    #Initialize with ones
    mask = ones(Float64,size(EPI_dense));
    mask[abs.(EPI_dense-EPI_sparse).!=0]=0
    # Data
    EPI_masked = EPI_sparse.*mask;
    stopFactor = 0.005; # The highest coefficient times
#stopFactor
    EPIinpainted50 = inpaint2D_accel(EPI_masked,mask,50,
stopFactor,shearletsystem);
    name_inpainted1 = split(name_dense,"/")
    name_inpainted1[4] = "Inpainted"
    last_name = name_inpainted1[5]
    last_name_split = split(last_name,"_")
    last_name_split[7] = "inpainted.png"
    last_name_split
    last_name = last_name_split[1]
    for i in 2:size(last_name_split)[1]
        last_name = last_name*"_"*last_name_split[i]
    end
    name_inpainted1[5] = last_name
    name_inpainted = name_inpainted1[1]
```

```
    for i in 2:size(name_inpainted1)[1]
        name_inpainted = name_inpainted*"/"*name_inpainted1[i]
    end
    name_inpainted
    Shearlab.imageplot(real(EPIinpainted50))
    savefig(name_inpainted, dpi = 80*3,bbox_inches="tight")
    clf()
end

# Inpaint the whole list
for i in 1:length(sparse_list)
    file_sparse = sparse_list[i];
    file_dense = dense_list[i];
    inpaint_file(file_sparse,file_dense);
end
```

# Appendix D

# Code for computing the depth map

The following code was used to detect the lines in the inpainted EPIs and compute the depth with the slope of the detected lines. We used the OpenCV implementation of Canny edge detector and Hough Line transform.

```
#! /usr/bin/python
# -*- coding: utf-8 -*-

# Import the libraries to use
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image

# Importing the image
name_inpainted = '../Diagrams/results/Inpainted/673_10_
                          102_7_48_8_inpainted.png';
img = cv2.imread(name)
name_strip = '../Diagrams/results/EPIs/
                 673_10_102_4_48_8_strip.png';

# Reading the size of the image in pixels
[size_y,size_x] = np.shape(img[:,:,0])

# Setting up the img to comput the lines
img = cv2.imread(name_inpainted)
strip = cv2.imread(name_strip)
img_lines = cv2.imread(name_inpainted)
gray = cv2.cvtColor(img_lines,cv2.COLOR_BGR2GRAY)
# Compute edges in the image with Canny edge detector
edges = cv2.Canny(gray,50,150,apertureSize = 3)
# Compute image lines with Hough transform
lines = cv2.HoughLines(edges,1,np.pi/180,200)
for line in lines:
    rho,theta = line[0]
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
```

```
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))
    if (x2-x1)!=0:
        if (y2-y1)/(x2-x1)!=0:
            cv2.line(img_lines,(x1,y1),(x2,y2),(0,0,255),2)

plt.rcParams["figure.figsize"] = [12,9]
plt.imshow(img_lines)
plt.show()

# Function to compute the slope
def slope(lines,i,size_x,size_y):
    line = lines[i]
    rho,theta = line[0]
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))
    # Find the slope
    slope = abs((y2-y1)/(x2-x1))*(size_x/size_y)*(1.1/683)
    return slope,x1,x2,y1,y2

# Example of line slope computation and visualization
img = cv2.imread(name_inpainted)
i = 15
slopei,x1,x2,y1,y2 = slope(lines,i,size_x,size_y);
cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2);
plt.imshow(img)
plt.rcParams["figure.figsize"] = [12,9]
plt.imshow(strip)
plt.show()
```

# Bibliography


[1] R.C. Bolles, H.H. Baker, D. H. Marimont, *Epipolar-plane image analysis and approach to determining structure from motion*, International Journal of Computer Vision, 1:7-55, 1987.

[2] G. Kutyniok, M. Genzel, *Asymptotic analysis of inpainting via universal shearlet Systems*, SIAM Journal on Imaging Sciences, 7(4), 2301-2339, 2014

[3] S. Vagharshakyan, R. Bregovic, A. Gotchev, *Light field reconstruction using shearlet transform*, IEEE Transactions on Pattern Analysis and Machine Intelligence, P(99), 2017

[4] E.H. Aderson and J.R. Bergen, *The plenoptic function and the elements of early vision*, Vision and Modeling Group, MIT Media Laboratory, MIT, 1991

[5] C.-K. Liang, Y.-C- Shih, H.Chen, *Light field analysis for modeling image formation*, IEEE Trans. Image Processing, 20(2), 446-460, 2011

[6] C. Kim, *3D Reconstruction and Rendering from High Resolution Light Fields*, Diss. ETH No. 22933, 2015

[7] H. Ives, *Parallax Stereogram and Process of Making Same* US patent 725, 567 1903

[8] G. Lippmann, *La Photographie Intégrale*, Academie des Sciences 146, 446-451, 1908

[9] E.H. Adelson, J. R. Bergen, *The plenoptic function and the elemnts of early vision*, Computational Models of Visual Processing, pages 3-20, 1991

[10] C. Tomasi, *Early Vision*, Encyclopedia of Cognitive Science, Level 2, 2006

[11] K. Marwah, G. Wetzstein, Y. Bando, R. Raskar, *Compressive Light Field Photography using Overcomplete Dictionaries and Optimized Projections*, ACM Transactions on Graphics (SIGGRAPH), 32(4), 2013

[12] C. Perwass, L. Wietzke, *Single lens 3D-camera with extended depth-of-field*, Human Vision and EleBuckheit and Donoho (1995)maging 2012, 829108, 2012

[13] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, P. Hanrahan, *Light field photography with a hand-held plenoptic camera*, Technical Report CSTR 2005-2, Stanford University, 2005

[14] E.H. adelson, J. Y. A. Wang, *Single lens stereo with a plenoptic camera*, IEEE International Conference on Computer Vision, 2007

[15] N. Joshi, W. Matusik, S. Avidan, *Natural video matting using camera arrays*, ACM Transactions on Graphics, 25(3), 779-786, 2006

[16] A. Veeraraghavan, R. Raskar, A. K. Agrawal, A. Mohan and J. Trumblin, *Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing*, ACM Transactions on Graphics, 26(3), 1-69, 2007

[17] G. Wetzstein, I. Ihrke, W. Heidrich, *On plenoptic multiplexing and reconstruction*, International Journal of Computer Vision, 101(2), 384-400, 2013

[18] M. Levoy, P. Hanrahan, *Light field rendering*, Proceedings of ACM SIGGRAPH, 31-42, 1996

[19] S. J. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen, *The Lumigraph*, Proceedings of ACM SIGGRAPH, 43-54, 1996

[20] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, M. Gross, *Scene reconstruction from high spatio-angular resolution light fields*, ACM Trans. Grpah, 32(4), 73:1-73:2, 2013

[21] A. Isaksen, L. McMillan, S.J. Gortler, *Dynamically Reparameterized Light Fields*, ACM SIGGRAPH, ACM Press, 297-306, 2000

[22] B. Javidi, F. Okano, *Three-Dimensional Television, Video and Display Technologies*, Springer-Verlag, 2012

[23] M. Levoy, R. Ng, A. Adams, M. Footer, M. Horowitz, *Light Field Microscopy*, ACM Transactions on Graphics, Proceedings of SIGGRAPH, 25(3), 2006

[24] N. C. Pégard, H. Y. Liu, N. Antipa, M. Gerlock, H. Adesnik, L. Waller, *Compressive light-field microscopy for 3D neural activity recording*, Optica 3, 5, 517-524, 2016

[25] R. Raskar, A. Agrawal, C. Wilson, A. Veeraraghavan, *The Discrete Focal Stack Transform*, Proc. ACM SIGGRAPH, 56 2008

[26] R. C. Bolles, H. H. Baker, *Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion*, International Journal of Computer Vision, 1, 7-55, 1987

[27] R. Gupta, R. I. Hartley, *Linear pushbroom cameras*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(9), 963-975, 1997

[28] G. Bradski, A. Kaehler, *Learning OpenCV*, O'Reilly Media, 2008

[29] R. Hartley, A Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2004

[30] C. Kim, *3D Reconstruction and Rendering from High Resolution Light Fields*, PhD Thesis of ETH Zurich, 2015

[31] K. Wolff, C. Kim, H. Zimmer, C. Schroers, M. Botsch, O. Sorkine-Hornung, A. Sorkine-Hornung, *Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction*, 3D International Conference on 3D Vision (3DV), 2016

[32] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, M. Gross, *Scene Reconstruction from High Spatio-Angular Resolution Light Fields*, ACM SIGGRAPH, 2013

[33] T. Basha, S. Avidan, A. Sorkine-Hornung, W. Matusik, *Structure and Motion from Scene Registration*, IEEE Conference on Computer Vision Pattern Recognition (CPVR) 2012

[34] D. G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, 60, 91, 2004

[35] D. Vernon, *Machine Vision*, Prentice-Hall, p. 98-99, 214, 1991

[36] C. Harris, M. Stephens, *A combined corner and edge detector*, In Proc. of Foruth Alvey Vision Conference, 147-151, 1988

[37] J. Shi, C. Tomasi, *Good Features to Track*, Vision and Pattern Recognition (CVPR94)i, 593-600, 2004

[38] B. D. Lucas, T. Kanade, *An iterative image registration technique with an application to stereo vision*, Proceedings of Imaging Understanding Workshop, 121-130, 198, 1981

[39] R. Priemer, *Introductory Signal Processing*, World Scientific, p.1., 1991

[40] J.B. Joseph Fourier, *Théorie analytique de la chaleur*, Paris: Firmin Didot, père et fils, 1822

[41] K. Gröchenig, *The mystery of Gabor frames*, J. Fourier Anal. Appl., 20(4): 865-895, 2014

[42] P. Goupillaud, A. Grossman, J. Morlet, *Cycle-octave and related transforms in seismic signal analysis*, Geoexploration, 23:85, p. 102, 1984

[43] S. Mallat, *A wavelet tour of signal processing: The Sparse Way*, Elsevier, 2009

[44] , G. Kutyniok, D. Labate, *Introduction to Shearlets*, Shearlets: Multiscale analysis for multivariate data, Eds. Birkhuser Boston, pp. 1-38, 2012

[45] G. Kutyniok, J. Lemvig, W.-Q. Lim, *Shearlets and optimally sparse approximation*, Shearlets: Multiscale analysis for multivariate data, Ed. Birkhuser Boston, pp. 145-197, 2012

[46] D.L. Donoho, *Sparse components of images and optical atomic decompositions*, Constr. Approx. 17(3), pp. 353-382, 2001

[47] E. Candès and D. Donoho, *Curvelets - a surprisingly effective nonadaptive*, Curves and Surface Fitting: Saint Malo, pp. 105-120, 1999

[48] K. Guo, G. Kutyniok, D. Labate, *Sparse multidimensional representations using anisotropic dilation and shear operators*, Nashboro Press, TN, pp. 189-201, 2006

[49] C. Fefferman, *A note on spherical summation multipliers*, Israel Journal of Mathematicas, 15, pp. 44-52, 1973

[50] G. Kutyniok, *Functional Analysis III: Lecture Notes*, Sommersemester 2016, TU Berlin, 2016

[51] G. Kutyniok, W.-Q. Lim, R. Reisenhofer, *Shearlab 3D: Faithful Digital Shearlet Transforms Based on Compaclty Supported Shearlets*, ACM Trans. Math. Software 42, Article No. 5 2016

[52] W.-Q. Lim, *Nonseparable shearlet transform*, IEEE Transactions on Image Processing, 2285, pp. 2056-2065, 2013

[53] I. Daubechies, *Ten lectures on wavelets*, volume 62 of CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 1992

[54] K. Guo and D. Labate, *Optimally sparse multidimensional representation using shearlets*, SIAM J. Math. Anal. 39, pp. 298-318, 2007

[55] G. Kutyniok, J. Lemvig, W.-Q. Lim, *Optimally sparse approximations of 3D functions by compactly supported shearlet frames*, SIAM Journal on Mathematical Analysis, 44(4), pp. 2962-3017, 2012

[56] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, J. Verdera, *Filling-in by joint interpolation of vector fields and gray levels*, IEEE Trans. Image Process. 10, pp. 1200-1211, 2001

[57] M. Elad, J.-L. Starck, P. Querre, D. L. Donoho, *Simultaneous cartoon and texture image inpainting using morphological component analysis*, Appl. Compt. Harmon. Anal. 19, pp. 340-358, 2005

[58] E. J. King, G. Kutyniok, W.-Q. Lim, *Image Inpainting: Theoretical Analysis and Comparison of Algorithms*, Proceedings of the SPIE, 8858, pp. 11, 2013

[59] E. J. King, F. Kutyniok, X. Zhuang, *Analysis of inpainting via clustered sparsity and microlocal analysis*, Math Imaging Vision, 48, pp. 205, 2014

[60] M.N. Do, M. Vetterli, *The finite ridgelet transform for image representation*, IEEE Trans. on Image Processing, 12(1), pp. 16-28, 2003

[61] J.-L. Starck, Y. Moudden, J. Bobin, M. Elad, and D.L. Donoho, *Morphological component analysis*, Proc. SPIE Optics and Photonics, 5914, pp. 59 140Q- 59 140Q-15, 2005

[62] J. Fadili, J.-L. Starck, M. Elad, and D. Donoho, *MCalab: Reproducible research in signal and image decomposition and inpainting*, Computing in Science Engineering, 12 (1), pp. 44-63, 2010

[63] T. Blumensath, M. Davies, *Normalised Iterative Hard Thresholding; guaranteed stability and performance*, IEEE J. Sel. Topics Signal Processing, 4(2), pp. 298-309, 2010

[64] R.O. Duda, P.E. Hart, *Use of the Hough Transformation to Detect Lines and Curves in Pictures*, Comm. ACM, 15, pp. 11-15, 1972

[65] P.E. Hart, *How the Hough Transform was Invented*, IEEE Signal Processing Magazine, 26(6), pp. 18-22, 2009

[66] P.V.C. Hough, *Method and means for recognizing complex patterns*, U.S. Patent 3, 069, 654, 1962

[67] T. Wiatowski, H. Boelcskei, *A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction*, CoRR, abs/1512.06293 (preprint), 2017

[68] B. K. Natarajan, *Sparse Approximate Solutions to Linear Systems*, SIAM J. Computing, 24, pp. 227-234, 1995

[69] J. Mairal, F. Bach, G. Ponce, G. Sapiro, *Online Dictionary Lerning For Sparse Coding*, International Conference on Machine Learning, 2009

[70] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, Y. Mori, *Depth estimation reference software (ders) 5.0*, ISO/IEC JTC1/SC29/WG11 M, 16923, 2009

[71] M. Tanimoto, T. Fujii, K. Suzuki, *View synthesis algorithm in view synthesis reference software 2.0 (vsrs2.0)*, ISO/IEC JTC1/SC29/WG11 M, 16090, 2009