AN O(N^2LOG(N)) PER PLANE FAST DISCRETE FOCAL STACK TRANSFORM

Fernando Pérez Nava[+], Jonás Philipp Lüke
[+]Departamento de Estadística, Investigación Operativa y Computación
Departamento de Física Fundamental y Experimental, Electrónica y Sistemas
Universidad de La Laguna
Email: fdoperez@ull.es, jpluke@ull.es

Abstract: This paper presents a new fast discrete focal stack transform that can be used in the 3D depth recovery problem using a single image of a scene from a plenoptic camera. A plenoptic camera uses a microlens array between the lens of the camera and the image sensor to measure the radiance and direction of all the light rays in a scene. The microlens array creates many small images taken from slightly different viewpoints, which can be used to obtain conventional images focused on a frontal plane of desired depth. The collection of such images is the focal stack of the scene. The focal stack is of great importance to the depth-from-focus approach since the elements in focus allow us to recover the depth of the objects in the scene. There are several approaches to the computation of the focal stack. Our approach is based on the Discrete Focal Stack Transform (DFST). We present a new algorithm that decreases the complexity of the DFST from O(n^3) per plane to O(n^2log(n)) per plane. This allows the computation of a focal stack with O(n/log(n)) times the number of planes of the DFST with the same computational complexity. From a practical point of view, in typical scenes we are able to increase more than 200 times the number of planes of the DFST. Finally we provide some computational results that show the validity of the approach.

1. Introduction

This work relates to image-based 3D reconstruction using a plenoptic camera. A plenoptic camera uses a microlens array between the lens of the camera and the image sensor to capture the 4D lightfield by measuring the radiance and direction of all the light rays in a scene. The conventional 2D images are obtained by 2D projections of the 4D lightfield. The fundamental ideas behind the use of plenoptic cameras can be traced back to the works of Lippmann and Ives on integral photography [1],[2]. These ideas have become practical with the development of the digital image processing and computer vision fields. One of the first plenoptic cameras based on the principles of integral photography was proposed in computer vision by Adelson and Wang [3] to infer depth from a single image. In their design, the plenoptic camera consisted of a camera body with a single main lens, a microlens array replacing the conventional camera sensor, and a relay lens to form the image on the sensor. More recently Ng et al. [4] presented a similar design, but produced in a portable hand-held device.

In this paper we will show a new technique to solve the 3D reconstruction problem by means of a single lightfield image from a plenoptic camera. Our approach to the 3D recovery problem is based on depth-from-focus approach. First, we process the lightfield image to obtain the focal stack i.e. to produce conventional photographs focused at frontal planes of different depths in the scene, and then we examine the elements in focus to recover the 3D depth of the scene. Our main contribution is related to the refocusing process whose goal is to obtain the focal stack. Given the lightfield of a scene, it is theoretically possible to obtain a conventional photograph focused at a desired depth by means of the Photography integral operator [4]. To discretize this operator it is necessary to solve two problems: to interpolate the lightfield and to approximate the integration process. The brute-force approach would interpolate the lightfield by nearest neighbour and approximate the integral by sums. If we assume that the plenoptic camera is composed of $n \times n$ microlenses and that each microlens generates a $n \times n$ image, the brute-force approach would need $O(n^4)$ operations to generate a photograph refocused on a determined plane. A significant improvement to this performance was obtained in [4] with the "Fourier Slice Photography" (FSP) technique that reformulates the problem in the Fourier domain. The FSP method is based on the extraction of an appropriate dilated 2D slice in the 4D Fourier transform of the lightfield. In this case interpolation takes place in the 4D Fourier domain using a 4D Kaiser-Bessel filter of constant size. This decreases the computational cost of the discretization of the Photography operator to $O(n^2 \log(n))$ operations to generate a refocused photograph. Another approach to the computation of the focal stack is the Discrete Focal Stack Transform (DFST) [5]. Instead of using nearest neighbour interpolation in the 4D spatial domain or Kaiser-Bessel interpolation in the 4D frequency domain, the DFST uses 4D trigonometric interpolation in the spatial domain and a discretization of the integral operator. The DFST has several desirable properties [5]: algebraic exactness, geometric exactness and parallelism with continuum theory. However, the use of an interpolation kernel whose size is $O(n^2)$ increases its computational complexity to $O(n^3)$ per refocused photograph. In this paper we present a new algorithm that decreases the complexity of the DFST from $O(n^3)$ per plane to $O(n^2 \log(n))$ per plane. This allows the computation of a focal stack with $O(n/\log(n))$ times the number of planes of the DFST with the same computational complexity. The main benefit of this increment in the number of refocused photographs in the focal stack is a 3D description of the scene with $O(n/\log(n))$ times more depth values than the previous DFST approach. Other approximate approaches to the computation of the focal stack include the Fast Approximate Focal Stack Transform (FAFST) [6] with $O(n^3)$ per plane.

This paper is divided in five sections. In Section 2 we introduce the DFST based focal stack. In Section 3 we present our new algorithm to compute the DFST. Section 4 shows how to solve the 3D reconstruction problem using the focal stack by means of a laplacian-based focusing measure. Section 5 contains some experimental results and Section 6 includes conclusions and future work.

2. The Discrete Focal Stack Transform (DFST)

In this section we present the Continuous Focal Stack Transform (CFST) which is based on the Photography Transform and the DFST as a discrete approximation of the CFST.

2.1. Definition of the Continuous Focal Stack Transform

To introduce the Photography transform we parameterize the lightfield defined by all the light rays inside a camera. We will use the two-plane parameterization of this lightfield and write $L_F(\boldsymbol{x},\boldsymbol{u})$ as the radiance travelling from position $\boldsymbol{u}=(u_1, u_2)'$ (apostrophe means transpose) on the lens plane to position $\boldsymbol{x}=(x_1, x_2)'$ on the sensor plane. F is the distance between the lens and the sensor (see figure 1 adapted from [4]).

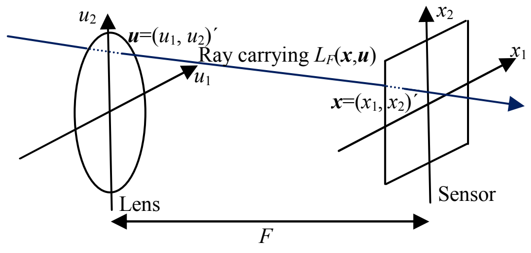
u_2
$\boldsymbol{u}=(u_1, u_2)'$
Ray carrying $L_F(\boldsymbol{x},\boldsymbol{u})$
u_1
x_2
x_1
$\boldsymbol{x}=(x_1, x_2)'$
Lens
Sensor
F

Figure 1 - Two plane parameterization of the lightfield.

The lightfield L_F can be used to compute conventional photographs at any depth αF. Let $\mathcal{P}_\alpha$ be the operator that transforms a lightfield L_F at a sensor depth F into a photograph formed on a film at sensor depth αF, then we have [4]:

$$\mathcal{P}_\alpha[L_F](\boldsymbol{x}) = \frac{1}{\alpha^2 F^2}\int L_F\left(\boldsymbol{u}\left(1-\frac{1}{\alpha}\right)+\frac{\boldsymbol{x}}{\alpha},\boldsymbol{u}\right)d\boldsymbol{u}. \tag{1}$$

This equation explains how to compute photographs at different depths from the lightfield L_F. When we compute the photographs for every sensor depth αF we obtain the Focal Stack transform $\mathcal{S}$ of the lightfield defined as [5]:

$$\mathcal{S}[L_F](\boldsymbol{x},\alpha) = \mathcal{P}_\alpha[L_F](\alpha\boldsymbol{x}) \tag{2}$$

2.2. The Discrete Focal Stack Transform (DFST).

In practice, the continuous formulation of the focal stack can not be used since a plenoptic camera only captures discrete samples of the lightfield. Discrete lightfields captured from a plenoptic camera are shown on figure 2 [4]. To obtain a discretized version of the focal transform we need to interpolate the lightfield L_F and to discretize the integral in (1).

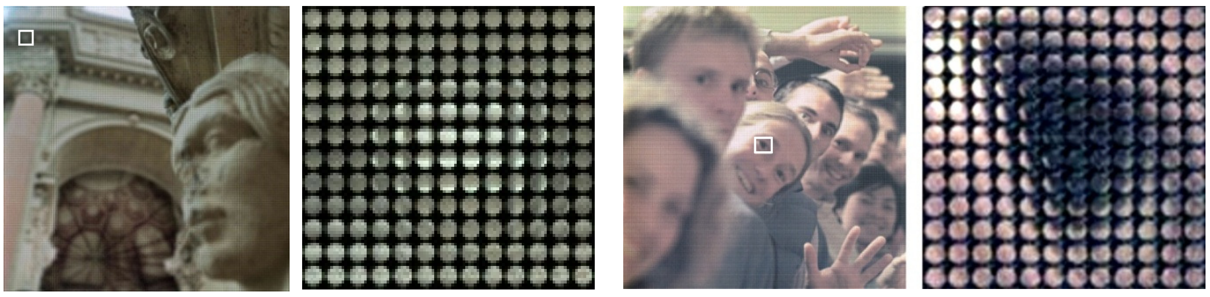

Figure 2 Plenoptic images and details from the white square in the plenoptic images [4].

To discretize the focal stack, the DFST supposes that lightfield L_F is periodic with periods $\boldsymbol{T_x}$ and $\boldsymbol{T_u}$ and that its values denoted by $L_F^d(\boldsymbol{x},\boldsymbol{u})$ are known for the set of points on a grid $\boldsymbol{G}$: $\boldsymbol{G} = \{(\hat{\boldsymbol{x}}\,\Delta x, \hat{\boldsymbol{u}}\,\Delta u)\}$, $\hat{\boldsymbol{x}} = -\boldsymbol{n_x} \dots \boldsymbol{n_x} \in \mathbb{Z}^2, \hat{\boldsymbol{u}} = -\boldsymbol{n_u} \dots \boldsymbol{n_u} \in \mathbb{Z}^2$,

which is a compact notation for a 4D grid where:

$\hat{x}_i = -n_x \dots n_x \in \mathbb{Z}, \hat{u}_i = -n_u \dots n_u \in \mathbb{Z}$, i=1,2, $\Delta x = \frac{T_x}{N_x}, \Delta u_i = \frac{T_u}{N_u}$, and:

$\boldsymbol{N_x} = (N_x, N_x)', \boldsymbol{N_u} = (N_u, N_u)', \boldsymbol{N_x} = 2\boldsymbol{n_x}+1, \boldsymbol{N_u} = 2\boldsymbol{n_u}+1,$

$\boldsymbol{n_x} = (n_x, n_x)', \boldsymbol{n_u} = (n_u, n_u)'.$

Then, L_F^d is defined as the trigonometric polynomial that interpolates the points in $\boldsymbol{G}$:

$$L_F^d\,(\boldsymbol{x},\boldsymbol{u}) = \sum_{\widehat{\boldsymbol{a}}=-\boldsymbol{n}_x}^{\boldsymbol{n}_x} \sum_{\widehat{\boldsymbol{b}}=-\boldsymbol{n}_u}^{\boldsymbol{n}_u} C^d(\boldsymbol{a},\boldsymbol{b}) e^{2\pi i(\boldsymbol{x}'\boldsymbol{a}+\boldsymbol{u}'\boldsymbol{b})}\, |\Delta\boldsymbol{a}||\Delta\boldsymbol{b}|, |\Delta\boldsymbol{a}| = \frac{1}{|\boldsymbol{T}_x|}, |\Delta\boldsymbol{b}| = \frac{1}{|\boldsymbol{T}_u|}, \quad (3)$$

$$C^d(\boldsymbol{a},\boldsymbol{b}) = \sum_{\widehat{\boldsymbol{x}}=-\boldsymbol{n}_x}^{\boldsymbol{n}_x} \sum_{\widehat{\boldsymbol{u}}=-\boldsymbol{n}_u}^{\boldsymbol{n}_u} L_F^d\,(\boldsymbol{x},\boldsymbol{u})\, e^{-2\pi i(\boldsymbol{x}'\boldsymbol{a}+\boldsymbol{u}'\boldsymbol{b})}\, |\Delta\boldsymbol{x}||\Delta\boldsymbol{u}|,$$

where $|\boldsymbol{x}|$ stands for the product of all components in vector $\boldsymbol{x}$ and

$$\sum_{\boldsymbol{i}=-\boldsymbol{n}_i}^{\boldsymbol{n}_i} f(\boldsymbol{i}) = \sum_{i_1=-n_{i_1}}^{n_{i_1}} \sum_{i_2=-n_{i_2}}^{n_{i_2}} f(i_1,i_2). \quad (4)$$

Then, the DFST approximates the integral by sums and defines of the discrete Photography operator for $|\alpha|\Delta x \geq |1-\alpha|\Delta u$ as [5]:

$$\mathcal{P}_\alpha^d[L_F^d](\boldsymbol{x}) = \frac{1}{|\alpha|^2 F} \sum_{\widehat{\boldsymbol{u}}=-\boldsymbol{n}_u}^{\boldsymbol{n}_u} L_F^d\left(\boldsymbol{u}\left(1-\frac{1}{\alpha}\right)+\frac{\boldsymbol{x}}{\alpha},\boldsymbol{u}\right)|\Delta\boldsymbol{u}|, \boldsymbol{u} = \widehat{\boldsymbol{u}}\cdot\Delta\boldsymbol{u}. \quad (5)$$

This definition implies a discrete Fourier slice theorem [5] that is used to compute the DFST in terms of the (unaliased) Discrete Fractional Fourier Transform $\boldsymbol{F}_N^{\boldsymbol{\alpha}}$ (FrFT)[7] (6),which can be computed with the same complexity of the Discrete Fourier Transform (DFT).

Unaliased 1D FrFT

$$(F_N^\alpha[f])(a) = \sum_{x=-n}^{n} f(x) e^{-\frac{2\pi i\alpha x a}{N}}, \quad N = 2n+1$$

Unaliased 2D FrFT

$$(\boldsymbol{F}_N^{\boldsymbol{\alpha}}[f])(\boldsymbol{a}) = \sum_{\boldsymbol{x}=-\boldsymbol{n}}^{\boldsymbol{n}} f(\boldsymbol{x}) e^{-\frac{2\pi i\alpha_1 x_1 a_1}{N_1}} e^{-\frac{2\pi i\alpha_2 x_2 a_2}{N_2}}$$

$$\boldsymbol{\alpha} = (\alpha_1,\alpha_2)', \boldsymbol{N} = 2\boldsymbol{n}+1, \boldsymbol{n} = (n_1,n_2)', \quad (6)$$
$$\boldsymbol{a} = (a_1,a_2)', a_i = -n_i \dots n_i, \boldsymbol{x} = (x_1,x_2)'.$$

When $\alpha = 1$ in the 1D FrFT we obtain the usual 1D unaliased DFT and when $\boldsymbol{\alpha} = (1, 1)'$ in the 2D FrFT we obtain the usual 2D unaliased DFT. The algorithm to compute the DFST [5] is described below. Fourier transforms are made over variables not fixed.

DFST rendering algorithm
Input: Sampled lightfield $\hat{L}_F^d(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{u}})$ and N_x, N_u (both even).
$\hat{x}_i = -n_x \dots n_x - 1, n_x = N_x/2, \hat{u}_i = -n_u \dots n_u - 1, n_u = N_u/2.$
Output: $\hat{S}(\widehat{\boldsymbol{x}},\hat{p}) = \alpha^2 F^2 |\Delta\boldsymbol{x}| S^d[L_F^d](\boldsymbol{x},\alpha), \boldsymbol{x} = \widehat{\boldsymbol{x}}\Delta x$
$\left(1-\frac{1}{\alpha}\right) = \hat{p}\frac{\Delta x}{n_u \Delta u}\ \ -n_u \leq \hat{p} \leq n_u,$
Step 0
For every $\widehat{\boldsymbol{u}}$ pad with $\lceil n_u/2 \rceil$ zero rows under the image $\hat{L}_F^d(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{u}})$ and $\lceil n_u/2 \rceil + 1$ zero rows over $\hat{L}_F^d(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{u}})$. Also pad with $\lceil n_u/2 \rceil$ zero columns on the left of $\hat{L}_F^d(\widehat{\boldsymbol{x}},,\widehat{\boldsymbol{u}})$ and $\lceil n_u/2 \rceil + 1$ zero columns on the right. Extend $\widehat{\boldsymbol{u}}$ with zeros so that $\hat{L}_F^d(\hat{x}_1,\hat{x}_2,\hat{u}_1,\hat{u}_2)$=0 when $\hat{u}_1$=$N_u + 1$ or $\hat{u}_2$=$N_u + 1$. We obtain then the extended lightfield $L_F^{d;ext}(\widehat{\boldsymbol{x}},\widehat{\boldsymbol{u}})$ with size: $N_x^{ext} = N_x + 2\lceil n_u/2 \rceil + 1$, $N_u^{ext} = N_u + 1$, $n_u = N_u/2$,and $\boldsymbol{N}_x^{ext} = (N_x^{ext}, N_x^{ext})'$, $\boldsymbol{N}_u^{ext} = (N_u^{ext}, N_u^{ext})'$.

Step 1

For every fixed $\hat{\boldsymbol{u}}$ compute S1$(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})$= $(\boldsymbol{F}_{\boldsymbol{N}_x^{ext}}^{1,1}[L_F^{d;ext}(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})])(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})$

Step 2

For every fixed $\hat{\boldsymbol{a}}$ compute S2$(\hat{\boldsymbol{a}},\hat{\boldsymbol{p}}) = (\boldsymbol{F}_{\boldsymbol{N}_u^{ext}}^{-\frac{N_u^{ext}}{n_u N_x^{ext}}\hat{\boldsymbol{a}}}[\mathrm{S1}(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})])(\hat{\boldsymbol{a}},\hat{\boldsymbol{p}})$

Step 3

For every fixed $\hat{p}$ compute S3$(\hat{\boldsymbol{x}},\hat{p})$= $\boldsymbol{F}_{\boldsymbol{N}_x^{ext}}^{-1,-1}[\mathrm{S2}(\hat{\boldsymbol{a}},\hat{p},\hat{p})]\ (\hat{\boldsymbol{x}})$

Step 4

Build $\hat{S}(\hat{\boldsymbol{x}},\hat{p})$ extracting from S3$(\hat{\boldsymbol{x}},\hat{p})$ the $\hat{\boldsymbol{x}}$ values $-\boldsymbol{n}_x \leq \hat{\boldsymbol{x}} \leq \boldsymbol{n}_x - 1$ and finally divide $\hat{S}(\hat{\boldsymbol{x}},\hat{p})$ by $(N_x^{ext}N_u)^2$.

The computational complexity of the algorithm above is O(n^4log(n)) (dominated by Step 1) and generates n refocused photographs. It can be shown, however, that the diagonal values S2$(\cdot,\hat{p},\hat{p})$ can be generated in O(n^2) and that each different $\hat{p}$ set of size n costs O(n^2) [5]. Therefore the DFST can generate log(n) of such sets without changing the complexity for a total of $n$log(n) planes and O(n^3) complexity per plane.

3. A fast Discrete Focal Stack Transform (DFST)

In this section we present a new technique to compute the DFST that computes O(n^2) planes with the same global computational complexity of O(n^4log(n)) that presents the previous algorithm. This new algorithm is based on the following theorem:

Theorem

If $\hat{L}_F^d(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}}) = L_F^d(\hat{\boldsymbol{x}}\Delta x,\hat{\boldsymbol{u}}\Delta u)$ is a sampled lightfield over a grid G defined as: $\boldsymbol{x} = \Delta x\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}} = -\boldsymbol{n}_x \dots \boldsymbol{n}_x$, $\boldsymbol{u} = \Delta u\hat{\boldsymbol{u}}, \hat{\boldsymbol{u}} = -\boldsymbol{n}_u \dots \boldsymbol{n}_u$, and $\left(1-\frac{1}{\alpha}\right) = \hat{p}\Delta p, |\alpha|\Delta x \geq |1-\alpha|\Delta u$, $\hat{p} = -n_p \dots n_p \in \mathbb{Z}, N_p = 2n_p + 1, n_p = 2n_x n_u$, then for $\boldsymbol{t} = \Delta x\hat{\boldsymbol{t}}, \hat{\boldsymbol{t}} = -\boldsymbol{n}_x \dots \boldsymbol{n}_x$, the Photography operator P_α^d can be written as:

$$P_\alpha^d\left[L_F^d\right](\alpha\boldsymbol{t}) \tag{7}$$

$$= \frac{|\Delta\boldsymbol{u}|}{\alpha^2F^2}\frac{1}{|\boldsymbol{N}_x|}F_{N_p}^{-\left(\frac{\Delta u N_p \Delta p}{\Delta x N_x}\right)}\left(\left(\boldsymbol{F}_{\boldsymbol{N}_x}^{-1,-1}\left[\sum_{\hat{\boldsymbol{u}}=-\boldsymbol{n}_u}^{\boldsymbol{n}_u}(\boldsymbol{F}_{\boldsymbol{N}_x}^{1,1}[\hat{L}_F^d(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})])(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})\ \delta\!\left(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}},\hat{k}\right)\right]\right)(\hat{\boldsymbol{t}},\hat{k})\right)(\hat{\boldsymbol{t}},\hat{p})$$

$$\delta(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}},\hat{k}) = \begin{cases}1 & \hat{\boldsymbol{u}}'\hat{\boldsymbol{a}} = \hat{k}\\ 0 & \hat{\boldsymbol{u}}'\hat{\boldsymbol{a}} \neq \hat{k}\end{cases}$$

Transforms are made over variables not fixed. Even though the formula may seem complicated, the resulting algorithm is quite simple as we see below.

Fast DFST rendering algorithm

Input: Sampled lightfield $\hat{L}_F^d(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})$ and N_x, N_u (both even).

$\hat{x}_i = -n_x \dots n_x - 1, n_x = N_x/2, \hat{u}_i = -n_u \dots n_u - 1, n_u = N_u/2.$

Output: $\hat{S}(\hat{\boldsymbol{x}},\hat{p}) = \alpha^2F^2|\Delta\boldsymbol{x}|S^d[L_F^d](\boldsymbol{x},\alpha), \boldsymbol{x} = \hat{\boldsymbol{x}}\Delta x$

$\left(1-\frac{1}{\alpha}\right) = \hat{p}\Delta p,\ \Delta p = \frac{\Delta x}{\Delta u n_p^{ext}}, \hat{p} = -n_p^{ext} \dots n_p^{ext} \in \mathbb{Z},\ n_p^{ext} = 2n_x^{ext}n_u, n_x^{ext} = n_x + \lceil n_u/2\rceil$

Step 0

Pad $\hat{L}_F^d(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})$ as in the DFST algorithm and obtain the extended lightfield $L_F^{d;ext}(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})$.

Step 1
For every $\hat{\boldsymbol{u}}$ compute $\mathrm{S1}(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})$= $(\boldsymbol{F}^{1,1}_{\boldsymbol{N}_x^{ext}}[L_F^{d;ext}(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})])(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})$
Step 2
Initialize $\mathrm{S2}(\hat{\boldsymbol{a}},\hat{k})$=0, $\hat{k}=-n_p^{ext}\ldots n_p^{ext}\in\mathbb{Z}$
For every $\hat{\boldsymbol{a}}$, $\hat{\boldsymbol{u}}$ evaluate $\hat{k}=\hat{\boldsymbol{u}}'\hat{\boldsymbol{a}}$ and update:

$\mathrm{S2}(\hat{\boldsymbol{a}},\hat{k})$= $\mathrm{S2}(\hat{\boldsymbol{a}},\hat{k})$+ $\mathrm{S1}(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})$.

Step 3
For every $\hat{k}$ compute $\mathrm{S3}(\hat{\boldsymbol{x}},\hat{k})=(\boldsymbol{F}^{-1,-1}_{\boldsymbol{N}_x^{ext}}[\mathrm{S2}(\hat{\boldsymbol{a}},\hat{k})])(\hat{\boldsymbol{x}},\hat{k})$
Step 4
For every $\hat{\boldsymbol{x}}$ compute $\mathrm{S4}(\hat{\boldsymbol{x}},\hat{p})$= $F_{N_p^{ext}}^{-\frac{N_p^{ext}}{N_x^{ext}n_p^{ext}}}$ $[\mathrm{S3}(\hat{\boldsymbol{x}},\hat{k})]\ (\hat{\boldsymbol{x}},\hat{p})$, $N_p^{ext}=2n_p^{ext}+1$.
Step 5
Build $\hat{S}(\hat{\boldsymbol{x}},\hat{p})$ extracting from $\mathrm{S4}(\hat{\boldsymbol{x}},\hat{p})$ the $\hat{\boldsymbol{x}}$ values $-\boldsymbol{n}_{\boldsymbol{x}}\leq\hat{\boldsymbol{x}}\leq\boldsymbol{n}_{\boldsymbol{x}}-1$ and finally divide $\hat{S}(\hat{\boldsymbol{x}},\hat{p})$ by $(N_x^{ext}N_u)^2$.

The computational complexity of the algorithm above is O(n^4log(n)) decomposed as follows: Step 1 computes n^2 DFTs of size n^2 so its complexity is O(n^4log(n)). Step 2 computes O(n^4) sums. Step 3 computes n^2 DFTs of size n^2 so its complexity is O(n^4log(n)) and finally Step 4 computes n^2 FrFTs of size n^2 so its complexity is O(n^4log(n)). Since the algorithm generates O(n^2) refocused photographs, the complexity per plane is O(n^2log(n)).

In the next table we show a comparison of the computational complexity of several algorithms that compute the focal stack.

Algorithm	Global Complexity	Number of planes	Complexity per plane
Brute Force	O(n^4)	O(1)	O(n^4)
FSP [4]	O(n^4log(n))	O(n^2)	O(n^2log(n))
DFST [5]	O(n^4log(n))	O($n$log(n))	O(n^3)
FAFST [6]	O(n^4)	O(n)	O(n^3)
Fast DFST	O(n^4log(n))	O(n^2)	O(n^2log(n))

Table 1: Computational complexity comparison

We note the remarkable fact that both the Fast DFST and the FSP obtain the best computational complexity per plane, but the Fast DFST uses an interpolation kernel whose size is O(n^2) instead of the constant size kernel of the FSP.

3.1. Generating a subset of the focal stack.

In practical situations the number of planes of the proposed DFST is too high. For example, for the 3840x3840 image on figure 2 (left) it would generate a focal stack with more than 6000 planes. Usually the number of required planes is much lower. For example, current autostereoscopic displays only use 256 planes. It could be argued that methods that generate a smaller number of planes [5], [6] could be better for this task. However, the FASFT [6] and DFST algorithm in [5] generate only 15 planes for the images in figure 2. An interesting fact about the Fast DFST is its ability to compute a smaller number of planes using the capacity of the FrFT to compute a shorter segment of the transform [7] using the following result:

Proposition

The first S=2s+1values of the Fractional Fourier Transform $F_N^{\alpha}[f]$ of an array f of size N=2n+1, with n=rs can be written as:

$$\sum_{d=0}^{r-1} e^{-\frac{2\pi i \alpha d a}{N}} \left(F_S^{\alpha r \frac{S}{N}}[f^*(cr+d)]\right)(a) - s \le a \le s\, , f^*(cr+d) = \begin{cases} f(cr+d) & |cr+d| \le n \\ 0 & \text{other case} \end{cases}$$

and the computational complexity to obtain them is Nlog(S).

This proposition allows us to generate a subset of the DFST. The formulation for the Fast DFST that computes n_p planes is as follows:

Subset Fast DFST rendering algorithm
Input: Sampled lightfield $\hat{L}_F^d(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ and N_x, N_u (both even).
$\hat{x}_i = -n_x \ldots n_x - 1, n_x = N_x/2, \hat{u}_i = -n_u \ldots n_u - 1, n_u = N_u/2.$
Output: $\hat{S}(\hat{\boldsymbol{x}}, \hat{p}) = \alpha^2 F^2 |\Delta\boldsymbol{x}| S^d[L_F^d](\boldsymbol{x}, \alpha), \boldsymbol{x} = \hat{\boldsymbol{x}}\Delta x$
$\left(1 - \frac{1}{\alpha}\right) = \hat{p}\Delta p,\ \Delta p = \frac{\Delta x}{\Delta u n_p}, \hat{p} = -n_p \ldots n_p \in \mathbb{Z},\ n_p c = n_p^{ext}, c \in \mathbb{N}$
Step 0
Pad $\hat{L}_F^d(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ as in the DFST algorithm and obtain the extended lightfield $L_F^{d;ext}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$.
Step 1
For every $\hat{\boldsymbol{u}}$ compute S1$(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}})$= $(\boldsymbol{F}_{N_x^{ext}}^{1,1}[L_F^{d;ext}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})])(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}})$
Step 2
Initialize S2$(\hat{\boldsymbol{a}}, \widehat{qk}, \widehat{mk})$=0. $-n_p \le \widehat{qk} \le n_p, 0 \le \widehat{mk} \le c$
For every $\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}$ evaluate $\hat{k} = \hat{\boldsymbol{u}}'\hat{\boldsymbol{a}}$, $\widehat{qk} = \lfloor \hat{k}/c \rfloor, \widehat{mk} = mod(\hat{k}, c)$ and update:
S2$(\hat{\boldsymbol{a}}, \widehat{qk}, \widehat{mk})$= S2$(\hat{\boldsymbol{a}}, \widehat{qk}, \widehat{mk})$+ S1$(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}})$.
Step 3
For every $\hat{\boldsymbol{a}}$ compute S3$(\hat{\boldsymbol{x}}, \widehat{qk}, \widehat{mk}) = (\boldsymbol{F}_{N_x^{ext}}^{-1,-1}[$S2$(\hat{\boldsymbol{a}}, \widehat{qk}, \widehat{mk})])(\hat{\boldsymbol{x}}, \widehat{qk}, \widehat{mk})$
Step 4
For every $\widehat{mk}$ compute S4$\left(\hat{\boldsymbol{x}}, \hat{p}, \widehat{mk}\right) = F_{N_p}^{-\frac{N_p c}{N_x^{ext} n_p}}[$S3$(\hat{\boldsymbol{x}}, \widehat{qk}, \widehat{mk})]\ (\hat{\boldsymbol{x}}, \hat{p}, \widehat{mk}), N_p = 2n_p + 1.$
Step 5
For every $\hat{p}$ compute:

$$\text{S5}(\hat{\boldsymbol{x}}, \hat{p}) = \sum_{\widehat{mk}=0}^{c-1} e^{\frac{2\pi i \widehat{mk}\hat{p}}{N_x^{ext} n_p}} \text{S4}\left(\hat{\boldsymbol{x}}, \hat{p}, \widehat{mk}\right)$$

Step 6
Build $\hat{S}(\hat{\boldsymbol{x}}, \hat{p})$ extracting from S5$(\hat{\boldsymbol{x}}, \hat{p})$ the $\hat{\boldsymbol{x}}$ values $-\boldsymbol{n_x} \le \hat{\boldsymbol{x}} \le \boldsymbol{n_x} - 1$ and finally divide $\hat{S}(\hat{\boldsymbol{x}}, \hat{p})$ by $(N_x^{ext} N_u)^2$.

The computational complexity of the algorithm remains the same (dominated by step 1).

4. Depth from focus 3D reconstruction using the focal stack

Once the focal stack is built we will use Depth-from-focus (DFF) techniques [8] to recover depth. DFF is a 3D reconstruction method, based on applying a focus measure to a set of differently focused photos. For a particular element of the scene, the focus measure determines the focus setting at which the element is brought into optimal focus, which can then be related to depth, according to prior calibration. DFF is most commonly realized by varying the focus setting and holding all other lens settings fixed. Alternative schemes involve moving the object relative to the camera. One disadvantage of traditional DFF is that the scene must remain stationary while the images are captured with different lens settings. As explained in the previous sections, the use of a plenoptic camera allows us to obtain several focused images from a single lightfield image.

A variety of focus measures have been proposed, which are basically contrast detectors within a small spatial window in the image. We use a Laplacian based focus measure *FM*:

$$\left(FM\left[P_\alpha^d\left[L_F^d\right]\right]\right)(\alpha\boldsymbol{x}) = -\left|\left(P_\alpha^d\left[L_F^d\right] * D_{x_1}^2\right)(\alpha\boldsymbol{x})\right| - \left|\left(P_\alpha^d\left[L_F^d\right] * D_{x_2}^2\right)(\alpha\boldsymbol{x})\right|, \tag{8}$$

convolving each photograph in the focal stack with smoothed Laplacian kernels in the x_1 and x_2 directions. This collection of laplacian convoluted focal stack photographs is the laplacian focal stack (LFS) [5]. These convolutions can be efficiently implemented as a product in the frequency domain modifying Step 1 in all DFST-based algorithms as follows:

Step 1
For every $\hat{\boldsymbol{u}}$ compute $S1_{x_i}(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})= (\boldsymbol{F}_{\boldsymbol{N}_x^{ext}}^{1,1}[L_F^{d;ext}(\hat{\boldsymbol{x}},\hat{\boldsymbol{u}})])(\hat{\boldsymbol{a}},\hat{\boldsymbol{u}})\ (\boldsymbol{F}_{\boldsymbol{N}_x^{ext}}^{1,1}[D_{x_i}^2])\ (\hat{\boldsymbol{a}})$,
$\boldsymbol{F}_{\boldsymbol{N}_x^{ext}}^{1,1}[D_{x_i}^2](\hat{\boldsymbol{a}}) \propto \exp\left(-(\hat{a}_1^2+\hat{a}_2^2)/\sigma^2\right)\hat{a}_i^2$, i=1,2.

The algorithm proceeds as before obtaining $\hat{S}_{x_i}(\hat{\boldsymbol{x}},\hat{p})$ i=1,2 and we finally compute the focus measure $FM\ (\hat{\boldsymbol{x}},\hat{p})=-\left(\left|\hat{S}_{x_1}(\hat{\boldsymbol{x}},\hat{p})\right| + \left|\hat{S}_{x_2}(\hat{\boldsymbol{x}},\hat{p})\right|\right)$.
We could greedily optimize the focus measure for each pixel independently: $\hat{p}(\hat{\boldsymbol{x}}) = \arg\min_{\hat{p}} FM\ (\hat{\boldsymbol{x}},\hat{p})$ but it gives better results to construct a prior favouring surface smoothness and to solve a regularized version of DFF. We use the LFS and the Markov Random Field (MRF) approach [9] to obtain the optimal depth minimizing an energy function [5]. This energy function is composed of a data energy E_d and smoothness energy E_s, $E = E_d + \lambda\ E_s$, where the parameter λ measures the relative importance of each term. The data energy is the sum of the per-pixel data costs $E_d = \sum_{\hat{\boldsymbol{x}}} FM\ (\hat{\boldsymbol{x}},\hat{p}(\hat{\boldsymbol{x}}))$. To define the smoothness energy we use a 4-connected neighborhood system, so the smoothness energy Es can be written $Es= \sum_{\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}} V_{\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}}(\hat{p}(\hat{\boldsymbol{x}}),\hat{p}(\hat{\boldsymbol{y}}))$ where $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ are 4-connected pixels. We use $V_{\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}}(\hat{p}(\hat{\boldsymbol{x}}),\hat{p}(\hat{\boldsymbol{y}})) = \min(\mu, |\hat{p}(\hat{\boldsymbol{x}}) - \hat{p}(\hat{\boldsymbol{y}})|^{\gamma})$ and optimize the energy function E using belief propagation [10].

5. Experimental Results

In this section we present some experimental results from the fast DFST. We have used the plenoptic images in figure 2. The left image has 3840×3840 pixels and is composed from an array of 256×256 microlenses. Each microlens generates a 15×15 image. The right image has 4380×4380 pixels and is composed from an array of 292×292 microlenses. Each microlens generates a 15×15 image. Intensity fall-off effects on the edges of microlens were compensated using a histogram-based non-linear transform. Pixels in the border of each microlens were discarded leading to 12x12 microlens images. In figure 3 we show a subset of images from the focal stack obtained from a detail of the plenoptic image on figure 2 (right) using the Subset Fast DFST algorithm. The scene gets focused from near objects (top left) to far objects (down right). In figure 4 we show a magnified version of three images from figure 3. It can be seen that the algorithm successfully focus on several planes in the scene. In figure 5 we show two images from the focal stack obtained from the plenoptic image on figure 2 (left) using the Fast DFST algorithm. Again we can see that the algorithm successfully focus on the face (left) and building (right). In figure 6 we present all-in-focus and depth images from the DFF technique and the focal stack computed with the fast DFST algorithm and plenoptic images in figure 2. The DFF technique is able to estimate the depth of the elements in the scene. Finally in figure 7 we see a 3D representation of the recovered 3D depth.

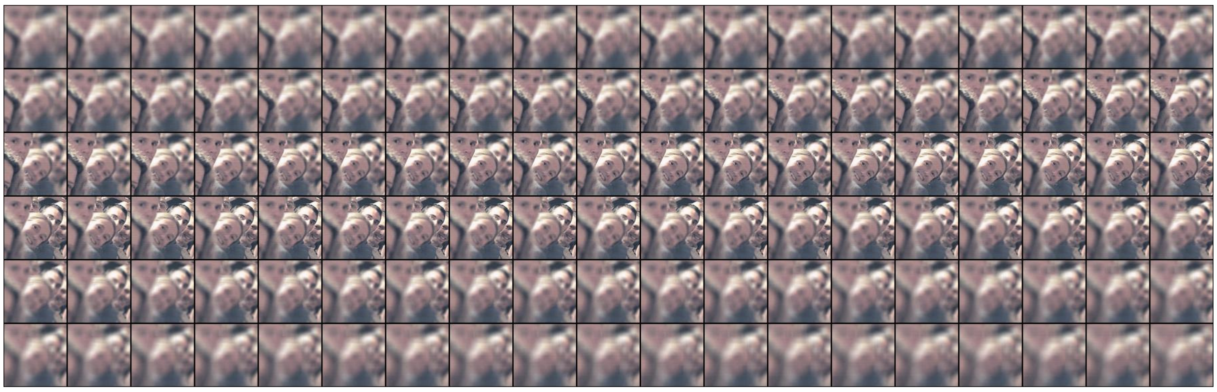
Figure 3 Subset of images from the focal stack obtained from a detail of the plenoptic image on figure 2 (right) using the Subset Fast DFST algorithm.

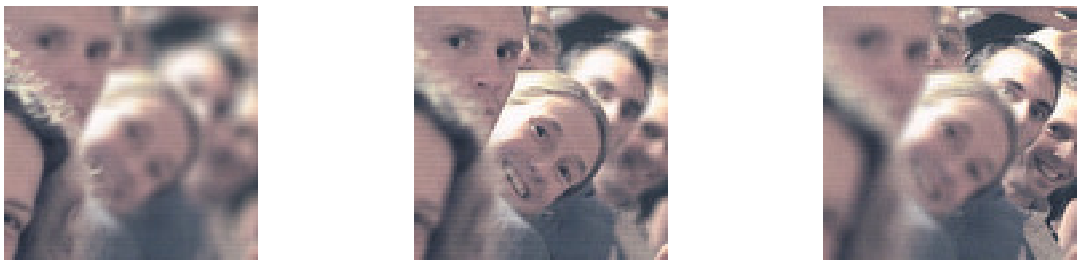
Figure 4 Images extracted from figure 3 and focused on several planes.

Figure 5 Images from the focal stack obtained from a detail of the plenoptic image on figure 2 (left) using the Fast DFST algorithm.

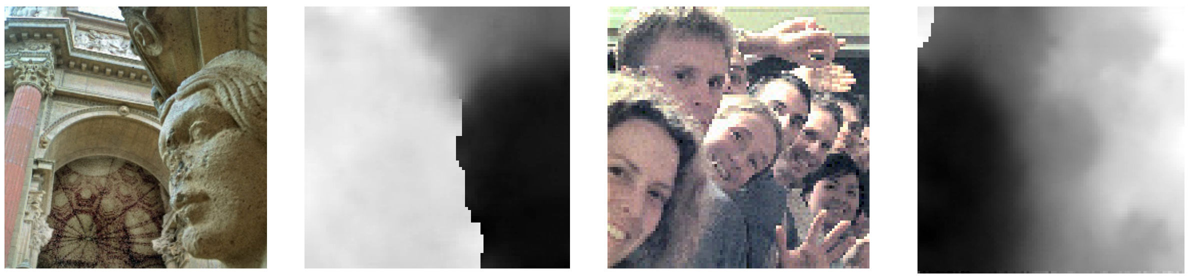
Figure 6 All-in-focus and depth images (grey levels represent depth) from the Depth from Focus technique and the focal stack computed with the fast DFST algorithm and plenoptic images in figure 2. Left and right images have 216 and 337 depth values respectively.

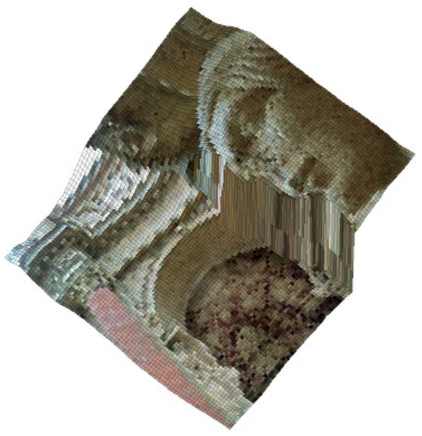
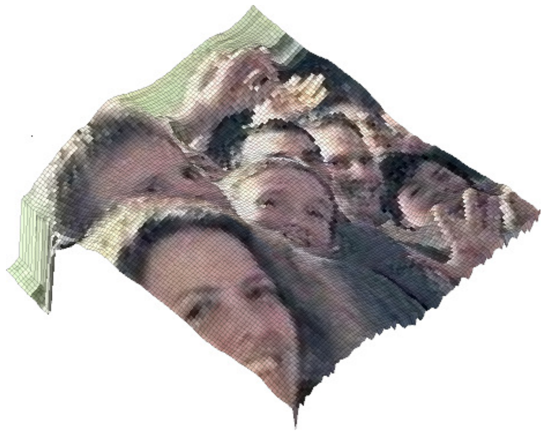

Figure 7 3D surface representation of depth with the all-in-focus image.

6. Conclusions

We have presented a new fast algorithm that decreases the complexity of the DFST from O(n^3) per plane to O(n^2log(n)) per plane. This allows the computation of a focal stack with O(n/log(n)) times the number of planes of the DFST with the same computational complexity. The use of depth from focus techniques allows us to recover depth using a single plenoptic image from a focal stack built with the fast DFST. Future improvements will consist on porting the technique to GPUs and FPGAs to obtain real-time processing.

Acknowledgements

This work has been funded by “Programa Nacional I+D+i” (Project DPI 2006-09726) of the “Ministerio de Ciencia y Tecnología”, and by the “European Regional Development Fund".

References:

[1] F. Ives. Patent US 725,567. 1903. 1

[2] G. Lippmann. Epreuves reversibles donnant la sensation du relief. Journal of Physics, 7(4):821–825, 1908.

[3] T. Adelson and J. Wang. Single lens stereo with a plenoptic camera. IEEE Transactions on Pattern Analysis and Machine Intelligence, pages 99–106, 1992.

[4] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, et al. Light field photography with a hand-held plenoptic camera. Computer Science Technical Report CSTR, Jan 2005.

[5] F. Pérez, J.G. Marichal, and J.M. Rodríguez-Ramos, The Discrete Focal Stack Transform, Proc. of the European Signal Processing Conference, 2008.

[6] J.G. Marichal, J.P. Lüke, F. Rosa, F. Pérez, and J.M. Rodríguez-Ramos, A Fast Approximate Discrete Focal Stack Transform, Proc. of the 3DTV Conference, 2009.

[7] D. H Bailey and P. N. Swartztrauber, The fractional Fourier transform and applications, SIAM Review 33(3), pp.389-404, (1991).

[8] S.K. Nayar Y. Nakagawa, Shape from focus**,** IEEE trans. on PAMI Vol. 16 nº 8 pp: 824 - 831, 1994.

[9] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, International Journal of Computer Vision, 47(1/2/3):7-42, 2002.

[10] P.F. Felzenszwalb; D.R., Huttenlocher, Efficient belief propagation for early vision,Computer Vision and Pattern Recognition, 2004. CVPR 2004. vol.1, pp. I-261-I-268, 2004.