National Observatory of Athens

University of Peloponnese
Department of Informatics and Telecommunications

National Center for Scientific Research "Demokritos"

M.Sc. in Space Science Technologies and Applications

Panoramic Images: Constructing Wide Field of View Digital Composites Using 2D Images or 4D Light-Fields.

Nikolaos K. Kalogeras

Supervisor:

Prof. Nikolaos Doukas

Athens

October 2020

Abstract

The creation of digital panoramic images, has come a long way in the last twenty years. The evolution of the algorithms and techniques we used to create 2D panoramic images (Composites), is a fascinating study by itself. We then examine the possible successor to 2D digital composite imaging, with the introduction of 4D lightfields and the ever evolving processes that can create panoramic lightfield images.

Picture above: Fool Moon - A 100,000 photos digital composite of the moon

Andrew McCarthy

https://www.instagram.com/cosmic_background

Contents

Panoramic Images:

Constructing Wide Field of View Composites with

2D Images or 4D Light-Fields

1. Introduction

Panoramic and lightfield photography methods are possibly different solutions, to a very old problem: we must correctly translate the real 3D world around us, into a limited 2D interpretation. Each method approaches the solution from a different angle: Panoramic photography tries to combine a series of still images with a minimum overlap captured in succession, lightfields on the other hand capture a whole set of light rays, and then generate different facets of the same scene, out of them.

In other words, panoramas are exploiting the width of the scene, whereas lightfields depend on the depth of it. It seems reasonable to assume that, if we can combine these two different methods, then we will explore a new approach to the solution to the problem of translating 3D space to a 2D plane.

In contrast to conventional digital images, digital light fields contain both spatial and directional information that can be used for synthetic refocusing, multi-perspective recording, depth-variant filtering, and much more. However, while plenoptic cameras are commercially available and plenoptic displays are becoming feasible, the development of algorithms that process light fields is lagging behind.

Nearly every modern digital camera or cell phone supports Digital Panorama imaging. However, the application of such well known image processing techniques to plenoptic data is not straightforward, as recorded spatial and directional information is neither independent nor can it be processed in the same way. Thus, we must develop new processing methods to support the handling of light-field data [BIB01].

Nevertheless, before we can examine the process of integrating these two methods, we will present a brief description of each method including its theoretical background.

2. Related work

We will present a general discussion of digital panorama and lightfield imaging, focusing more closely, on the algorithms related to each method.

Digital Panorama Stitching

[MRV80] describes a corner detector for the first time, from a sensor-captured image. Obstacle avoidance and real world navigation uses this method, and [CH88] describes a combination of both a corner and an edge detector. [DGL04] presented a method of selecting distinctive image features from scale invariant key points (SIFT), while [TL93] expanded the concept by detecting blob like image structures. [BTG09] presented a more optimized method of detecting features (SURF) and finally [SR05] and [SLW07], present methods of actually converting 2D images into Panoramas.

Lightfields

[MDY15] proposed a method for developing lightfields captured from an airborne sensor. Naive Multi Perspective Image Stitching [B07] addresses lightfield panorama creation while [BOB13] is proposing an alternate similar method based on focal stacks. Both of these methods use different 2D slices of lightfields, to create panoramas. [BIB01] and [GYKLY01] present alternative ways of creating lightfield panoramas, based on transforming light-ray coordinates.

3. Digital stitching of segmented panoramas

A **panorama** (formed from Greek πᾶν "all" + ὅραμα "sight") is any wide-angle view or representation of a physical space, whether in painting, drawing, photography, film, seismic images or a three-dimensional model.

Panoramic photography [PP12] is a technique of photography, using specialized equipment or software, which captures images with horizontally elongated fields of view. It is sometimes known as wide format photography. The term has also been applied to a photograph that is cropped to a relatively wide aspect ratio, like the familiar letterbox format in wide-screen video.

While there is no formal division between "wide-angle" and "panoramic" photography, "wide-angle" normally refers to a type of lens, but using this lens type does not necessarily make an image a panorama. An image made with an ultra-wide-angle fisheye lens covering the normal film frame of 1:1.33 is not automatically considered a panorama. An image showing a field of view approximating, or greater than, that of the human eye – about 160° by 75° – may be termed panoramic.

Fig. [1]

A 19th century panorama. "Wharf at Yonkers", watercolor, Samuel Colman's 1869-1870.

Fig. [2]

A panorama of Melbourne's Yarra River at twilight. Taken on 26 August 2005, by David Iliff, from Melbourne, Australia. The author used PTGui for stitching the image. Less daylight, and many reflections on the water, make this picture a very good example of digital stitching. Figures 1&2 display a common thematic thread, even though the crafting methods are vastly different.

This generally means it has an aspect ratio of 2:1 or larger, the image being at least twice as wide as it is high. The resulting images take the form of a wide strip. Some panoramic images have aspect ratios of 4:1 and sometimes 10:1, covering fields of view of up to 360 degrees. Both the aspect ratio and coverage of field are important factors in defining a true panoramic image.

Segmented panoramas, also called stitched panoramas, are constructed by joining multiple photographs with slightly overlapping fields of view to create a panoramic image. Stitching software is used to combine multiple images.

Ideally, in order to stitch images correctly together without parallax error, we must rotate the camera about the center of its lens entrance pupil. In digital photography, the most common method for constructing panoramas, is to take a series of pictures and stitch them together.

4. Overview

Section 5 describes 2D digital panoramic imaging, with each subsection examining the theory behind feature detection, image stitching and panorama projection. In section 6, we present real life examples and experiments. These panoramic images were created with commercial of-the-self equipment, and freely available open source software (Hugin).

Section 7 focuses in light-fields in general, while in section 8 we examine the more advanced methods of light-field panoramic imaging, with each subsection examining the theory of creating, and then describing the still ongoing research of creating 4D Panoramic Images, combining a large number of light fields. Finally, we present our concluding remarks in section 9.

5. Digital Image Stitching

i. *Feature Detection*

Feature detection, is the foundation of digital panoramas creation. Finding common features in a series of images creates the framework, which is vital for constructing them. Algorithms for discovering common features, have become more complex, over the years, covering topics from simple corner detection, all the way to, discovering more complicated image structures, like blobs [FDD14]:

Moravec's Corner Detector: Most of the time a corner in an image can be defined, as that part of the image, in which major variations in intensity occur. Moravec assumed an imaginary window, traversing the image, by a small amount in various directions, and in each shift of this window, the average changes of image intensity are determined [MRV80].

Harris – Stephens & Shi – Tomasi Corner Detectors: Harris & Stephens continued Moravec's work, by converting the simple idea behind it, to a mathematical form [CH88], [JS94]. They describe the imaginary window, and its task is to find the difference in intensity E(u,v) in the window, while it shifts in all directions.

Both detectors work in a similar way. In fact they are identical up to a point, and they differ in the utilization of the λ_1 and λ_2, λ_1 and λ_2 being the Eigen values of the array M (second-moment matrix), that is calculated from the mathematical expression of Moravec's Corner Detector. Both algorithms perform equally well, but Shi - Tomasi seems to be a little faster. However, now-days in a common desktop PC configuration, the difference in speed is negligible.

Scale Invariant Feature Transform (SIFT): Harris – Stephens & Shi – Tomasi Corner Detectors corner detectors are very robust, and are also rotation-invariant, provided that the scale of the image remains unchanged. Nevertheless, if we scale the image, it is deformed and a corner may not, remain a corner.

For addressing the issue in 2004, D.Lowe, developed a new approach called Scale Invariant Feature Transform (SIFT). The purpose of the new algorithm is to be robust and to solve problems regarding image rotation, scaling, affine deformation, viewpoint change, noise, and illumination changes [PPS13].

SIFT transforms image data into scale-invariant coordinates relative to local features. An important aspect of this approach is that it generates large numbers of features that densely cover the image over the full range of scales and locations. A typical image of size 500x500 pixels will give rise to about 2000 stable features (although this number depends on both image content and choices for various parameters).

The quantity of features is particularly important for object recognition, where it is a required ability to detect small objects reliably in cluttered backgrounds, under different viewing conditions, and after many iterations. At least three features must be correctly matched, from each object for reliable identification.

The SIFT algorithm functions by extracting keypoints and computing its descriptors [DGL04]. A SIFT keypoint is a circular image region

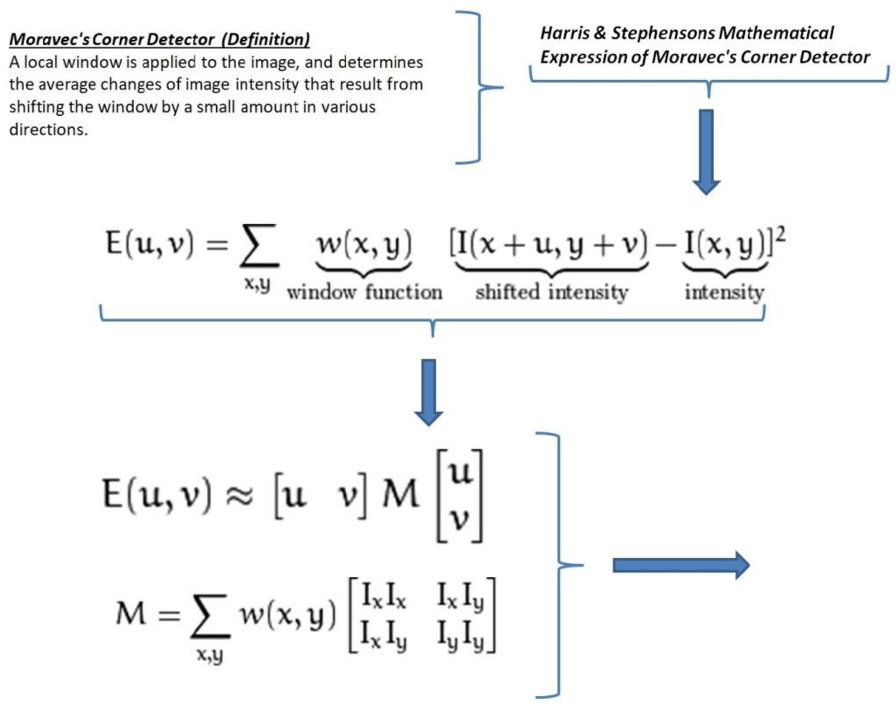

Moravec's Corner Detector (Definition)
A local window is applied to the image, and determines the average changes of image intensity that result from shifting the window by a small amount in various directions.
Harris & Stephensons Mathematical Expression of Moravec's Corner Detector
$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$
$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$
$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$

Fig. [3]
Evolution of an algorithm: Moravec described his Corner Detector Algorithm, and Harris - Stephenson transformed it into an equation. It basically finds the difference in intensity for a displacement of (u,v) in all directions.

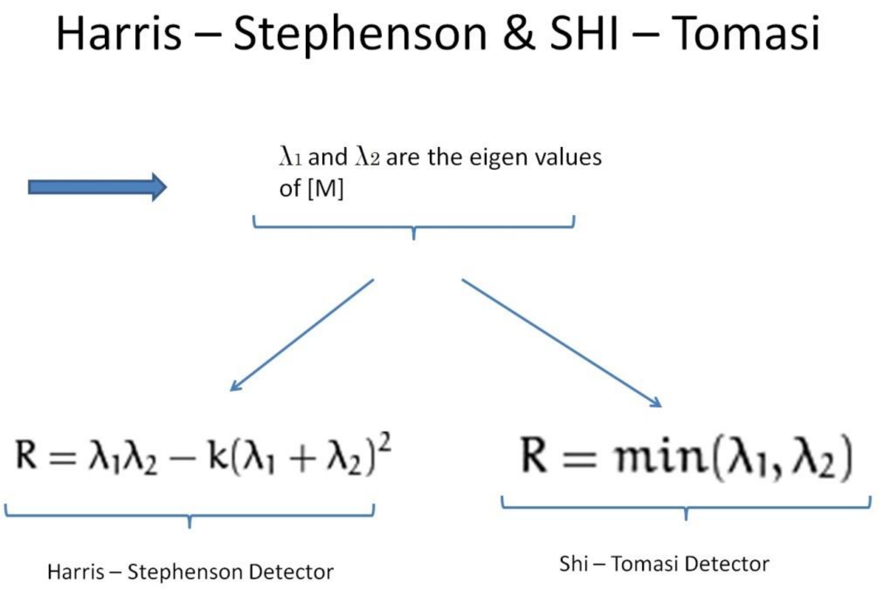

Harris – Stephenson & SHI – Tomasi
λ_1 and λ_2 are the eigen values of [M]
$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$
$R = \min(\lambda_1, \lambda_2)$
Harris – Stephenson Detector
Shi – Tomasi Detector

Fig. [4]
The different approaches of Harris - Stephenson and Shi - Tomasi. The utilization of λ1 and λ2, λ1 and λ2 being the Eigen values of the second-moment matrix M, is the main difference between these two algorithms.

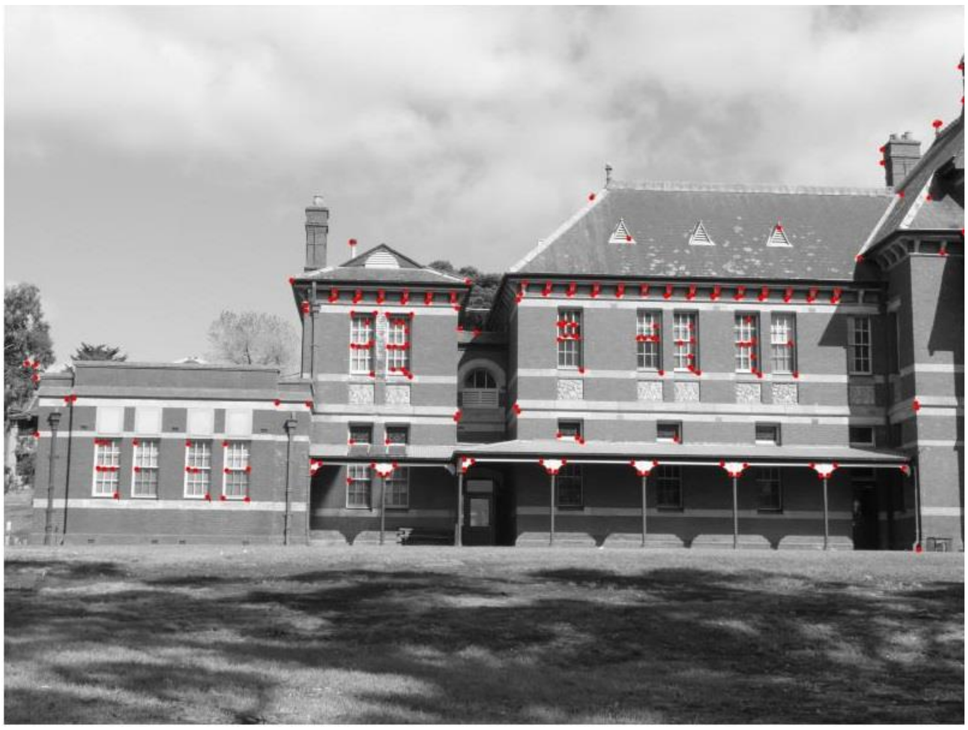


Fig. [5]
Corner Detection using Matlab and the Harris - Stephenson algorithm. ***corner(),*** *is a built in function of Matlab , that can perform corner detection, using both variations of the algorithm, as parameters.*

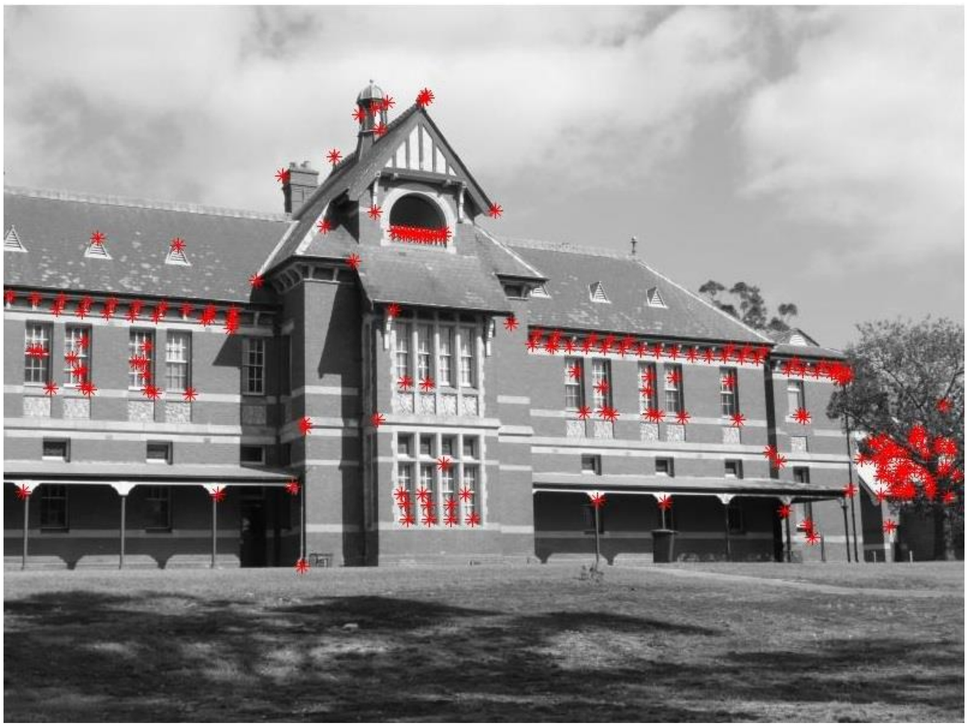


Fig. [6]
Corner Detection using Matlab and the Shi - Tomasi algorithm corner detection, using the same script as Fig [5]. Even though the algorithm perform, equally well in both cases, Shi - Tomasi is set as the default variation in the Matlab function. The difference in speed in a common desktop PC is negligible.

with an orientation. A geometric frame of four parameters describes it: the keypoint center coordinates x and y, its scale (the radius of the region), and its orientation (an angle expressed in radians).

Generally, blob-like image structures generate keypoints. A "blob" is a region in a digital image that differ in properties, such as brightness or color, compared to its surrounding regions. By searching for "blobs" at multiple scales and positions, the SIFT algorithm is invariant to translation, rotations, and rescaling of the image.

A SIFT descriptor is generated by computing image gradient magnitude and orientation at each image sample point in a region centered at key point [VVS12]. The major stages of computation used by SHIFT, to generate the set of image features [DGL04], are as follow:

(1) *Scale-space extrema detection.*

Scale - space representation is a framework for dealing with image structures, which naturally occur at different scales. From a given signal, we can generate a family of derived signals, according to the theory of this representation, by successively removing features when moving from fine to coarse scale [TL93]. In simple terms, the scale space is just a collection of images obtained, by progressively smoothing the input image, which is analogous to gradually reducing the image resolution. We call the smoothing level conventionally, scale of the image [SSI07].

Overall, the first stage of computation searches over all scales and image locations. It implements an efficient difference-of-Gaussian function, (a.k.a. the subtraction of one blurred version of an original image from another, less blurred version of the original. In the simple case of grayscale images, the blurred images are obtained by convolving the original grayscale images with Gaussian kernels having differing standard deviations [BSM20]), to identify potential interest points that are invariant to scale and orientation. (The desired effect is like reading maps of different scales, of the same geographic region. At larger scales, the larger geographical features are most prominent).

(2) *Keypoint localization*

We can find **keypoint** candidates, by comparing a pixel to its neighbors. The next step of the SIFT algorithm, is to perform a detailed fit to the nearby data (location, scale, etc.), to allow for points to

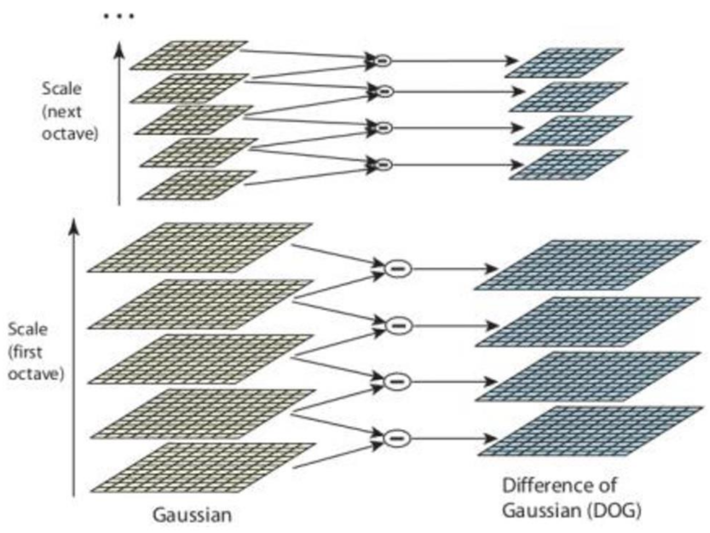


Fig. [10]
The idea behind Scale Space generation, by applying Difference of Gaussian (DoG): the subtraction of one blurred version of an original image from another, less blurred version of the original. After applying a Gaussian filter of different variance to the original image, the blurred versions are created.

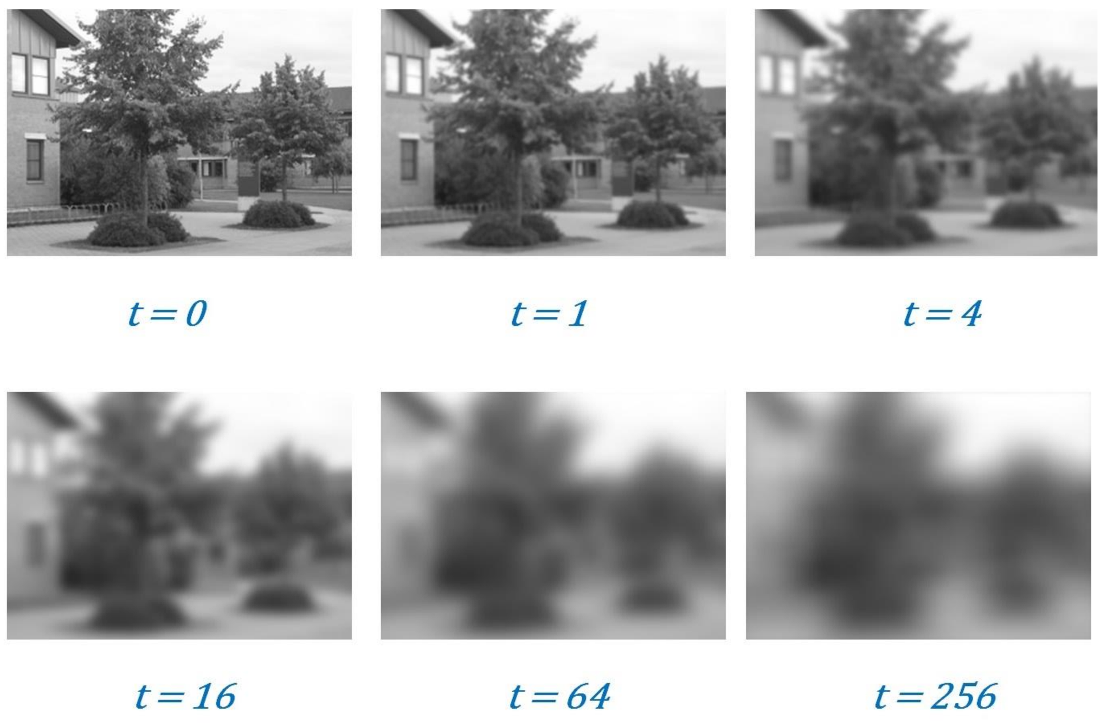


Fig. [9]
The actual generation of Scale Space, from a test image. Different scales t, are generated after filtering the image, with filters of different variance. In the bottom right sub image, regions with the same characteristics are prominent

be rejected, for low contrast (and are therefore sensitive to noise) or for being located on edges [SKF08].

(3) *Orientation assignment*

Image gradient, is the directional change in the intensity or color of the image. We sample magnitudes and orientations of image gradients, around the key point location [PPS13].

(4) *Keypoint descriptor*

By now, we have assigned every keypoint, a location in the image, a scale, and an orientation [DGL04]. The keypoint descriptor is computed using image gradient magnitude and orientation in a region centered at the keypoint. This is the key step in achieving invariance to rotation as the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. [IG10]

Finally, the descriptor vectors are matched between different images. We base the matching on a distance between the vectors, e.g. the Mahalanobis or Euclidean distance. The dimension of the descriptor has a direct impact on the time this takes, and a lower number of dimensions is therefore desirable since computation time is analog to the square of the vector dimension.

Speeded-Up Robust Features (SURF): In SIFT Lowe utilized Difference of Gaussian between the blurred images, for creating scale-space. However, even with this approach, scale-space generation is a time consuming process. Moreover the high dimensionality of the descriptor is a drawback of SIFT especially at the matching step.

In 2006, a speeded-up version of SIFT, a new algorithm by Bay, H., Tuytelaars, T. and Van Gool, L was introduced appropriate named, “SURF: Speeded Up Robust Features”. As name suggests, it is a speeded-up version of SIFT [BTG09].

In SIFT, Lowe approximated Laplacian of Gaussian with Difference of Gaussian (DoG) for finding scale-space. SURF goes a little further and approximates DoG with a Box Filter. One big advantage of this approximation is that, convolution with box filter can be easily calculated with the help of integral images, a data structure and algorithm for quickly and efficiently generating the sum of values in a rectangular subset of a grid. Moreover, it is possible to perform box filter convolution, in parallel for different scales.

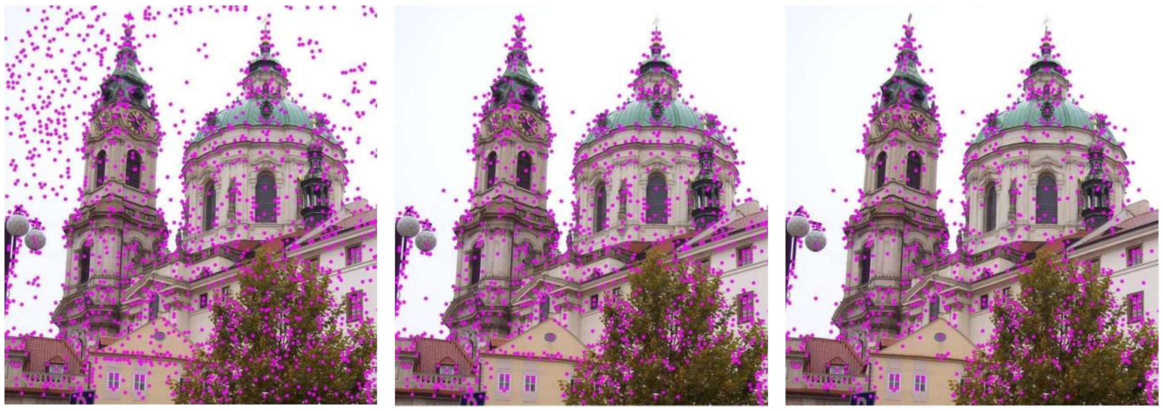


Fig. [11]
Keypoint localization and filtering.
Left : Location of possible keypoint candidates.
Center: Filtering low contrast and those located on the edges candidates.
Right: The algorithm will process the remaining keypoints, in the next step.

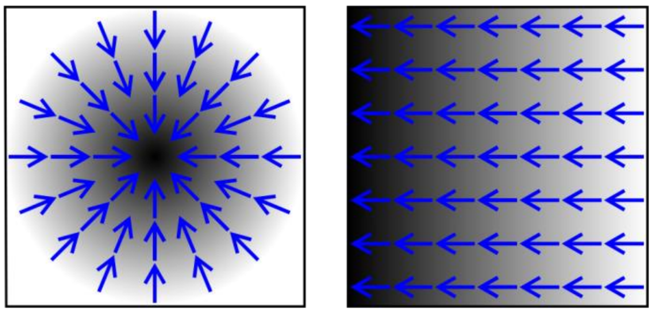


Fig. [12]
Keypoint descriptors orientation and generation. Gradients: Blue arrows indicate the direction of the gradient. Dark areas represent higher values.

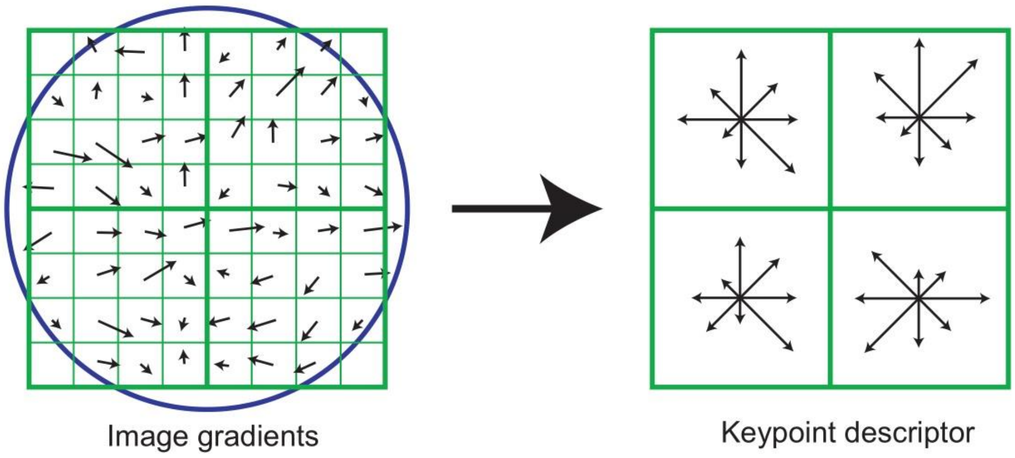


Fig. [13]
Keypoint descriptors orientation and generation. After computing image gradients , in a region around the keypoint location, a Keypoint descriptor is generated.
Right: A 2x2 descriptor array computed from an 8x8 set of samples.

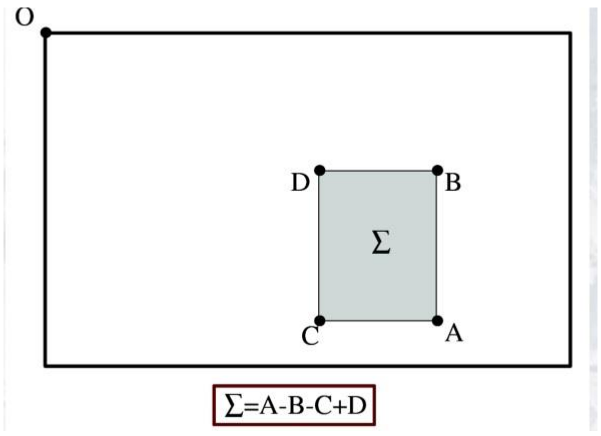


Fig.[14]
SURF reduces computation time significantly, since it calculates the sum of pixel intensities in a rectangular region. Only 3 additions needed, and calculation time is independent of the size [18].

Fig. [15]
SURF works like a blob detector. In the image above, it detects the white blobs on wings of the butterfly.

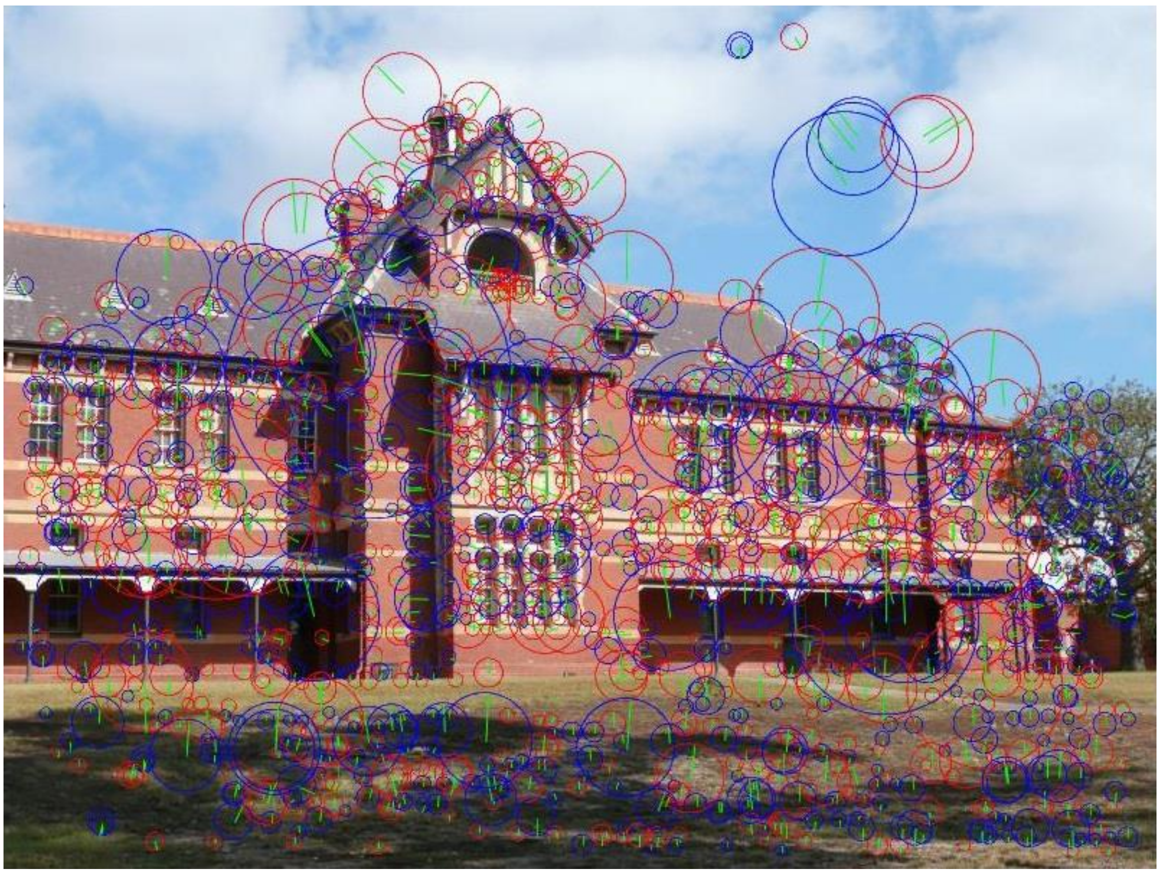


Fig. [16]
SURF detection, using the OpenSurf implementation in MatLab.

Circles represent interest points that are detected within the image. The size of them represent scales. Green lines represent orientation, and the color of the circles (red or blue), denotes bright blobs on dark backgrounds (Red), or dark blobs on bright backgrounds (Blue).

For the extraction of the descriptor, the first step consists of constructing a square region centered on the interest point, and oriented along the orientation selected by the descriptor. (If the image is upright, this transformation is not necessary). The region is split up regularly into smaller 4 × 4 square sub-regions. This keeps important spatial information in. For each sub-region, we compute a few simple features at 5×5 regularly spaced sample points.

d(x) is defined as, the Haar wavelet response in horizontal direction and *d(y)* the Haar wavelet response in vertical direction (filter size 2s). We define "horizontal" and "vertical" here in relation to the selected interest point orientation. To increase the robustness towards geometric deformations and localization errors, the responses *d(x)* and *d(y)* are first weighted with a Gaussian ($\sigma = 3.3s$) centered at the interest point.

Then, we sum up the wavelet responses *d(x)* and *d(y)* over each sub region and form a first set of entries to the feature vector. In order to bring in information about the polarity of the intensity changes, the sum of the absolute values of the responses, |*d(x)*| and |*d(y)*| is also extracted. This results in a descriptor vector for all 4×4 sub-regions of a standard length of 64.

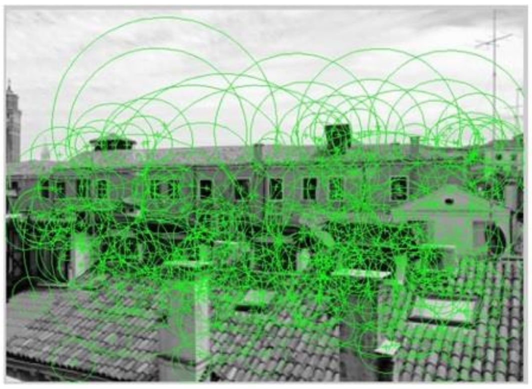
A Comparison of SIFT and SURF: [PPS13] has evaluated the previous two feature detection methods for features generation. Based on the experimental results, it is found that the SIFT has detected more number of features compared to SURF but it is suffered with speed. The SURF is fast and has nearly the same level of performance, as SIFT.

ii. *Image stitching*

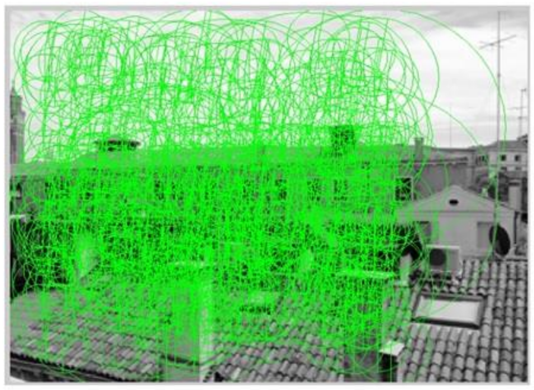
Image stitching or photo stitching, is the process of combining multiple photographic images with overlapping fields of view. Older techniques used to work by directly minimizing pixel-to-pixel dissimilarities (Direct Pixel Approach)[RS06]. Newer techniques work by extracting a sparse set of features and then matching these to each other, by implementing the algorithms and techniques discussed above.

Feature-based approaches have the advantage of being more robust against scene movement and are potentially faster, if we implement the technique in the right way. Their biggest advantage, however, is the ability to "recognize panoramas," i.e., to automatically discover the adjacency (overlap) relationships among an unordered set

(a) Original image1



(f) Detected features using SURF in image1



(c)Detected features in image1 using SIFT

Algorithm	Detected feature Points		Matching feature point	Feature matching Time
	Image1	Image2		
SIFT (Scale Invariant Feature Transform)	892	934	41	1.543 s
SURF (Speed Up Robust Feature)	281	245	28	0.546 s

Fig. [17]
SHIFT vs SURF comparison.

(a) The original image used as a "test bench". (f) Detected features using SURF. (c) Detected features using SHIFT. It's a trade off between speed and the number of features detected. SHIFT can reveal many more features than SURF does, but SURF can perform the same task as SHIFT at nearly half the time.

of images, which makes them ideally suited for fully automated stitching of panoramas taken by casual users [RS06][BD03]. Image stitching process consists of the following steps:

(1) Image Alignment and Registration

Image alignment in a pair of images (and in a series of images in general), is the process of discovering the appropriate mathematical model (motion model) that relates, the pixel coordinates of the first image with the pixel coordinates of second one. Once this model is discovered, pixel coordinates of second image, are transformed and introduced, into the coordinate system of the first image. This transformation of different sets of data from one coordinate system to another, using the motion model determined in the alignment phase, defines the image registration phase. Image registration, can be achieved with Direct Pixel and Feature-based methods.

Direct pixel approaches, shift or warp the images relative to each other and look at how much the pixels agree. Since is an older and computational demanding, direct pixel methods were superseded by feature-based approaches. With this method, distinctive features from each image are extracted, are matched together to establish an underlying association, and then estimate the geometric transformation between them. If these features are well distributed over the image, enough correlations between them are revealed, to permit stitching of the images.

(2) Compositing

Steps leading to final production of the stitched image, an image that we have now transformed into a mosaic or panorama, after the registration phase, include:

1. The selection of a final compositing surface (flat, cylindrical, spherical, etc.).
2. The reference image that will define the final view – perspective,
3. The selection of those pixels that contribute to the final composite in a way of optimally combining them to minimize visible seams, blur, and ghosting , an overall process known as blending.

Choosing a Compositing Surface – Projections: a natural approach for representing the final image, in case that only a

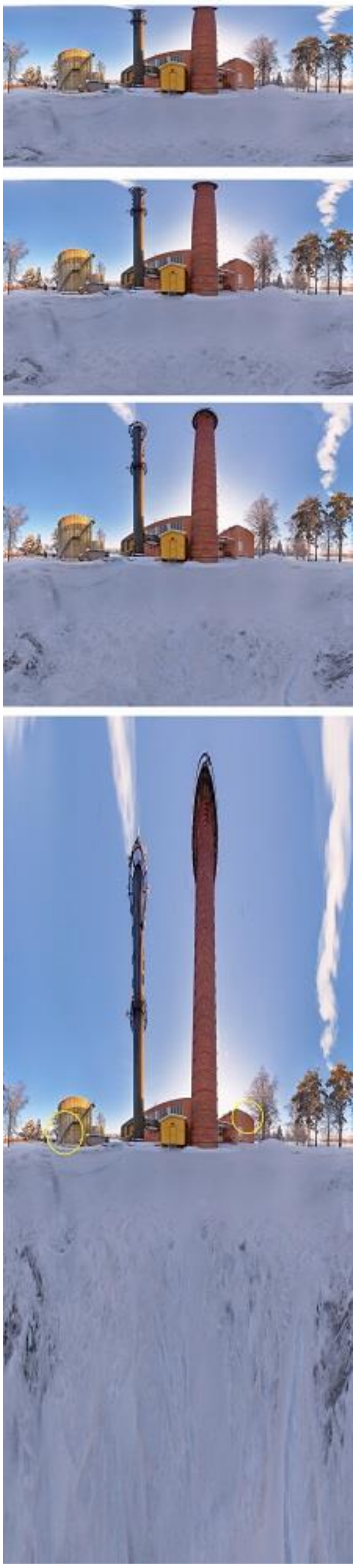


Fig. [18] Comparison of various cylindrical panoramic formats.

VFoV: 165 degrees

HFoV: 360 degrees.

There is no distortion in the vertical, but long straight lines are bended in the horizontal plane.

handful of images are stitched together, is to select one of these images as a reference and then warp all of the remaining images into the coordinate system of this reference image. We can the resulting composite a "flat panorama", since the projection onto the final surface is still a perspective projection, and hence straight lines remain straight (which is often a desirable attribute) [RS06].

For a larger number of images however, that result in a wider field of view composite, a flat representation cannot be maintained without excessively stretching pixels near the border of the image. (In practice, flat panoramas start to look severely distorted once the field of view exceeds 90 ◦ or so.) The usual choice for compositing larger panoramas is to use a cylindrical or spherical projection.

Projections: The choice of projection is somewhat application dependent, and involves a tradeoff between keeping the local appearance undistorted (e.g., keeping straight lines straight) and providing a reasonably uniform sampling of the environment:

Still Panoramic images, usually employ *planar/rectilinear* or *cylindrical projections* [BPG07]. A planar/rectilinear panorama, is displayed on a flat plane, and is usually stored as a single, flat-stitch rectilinear rendering image that can be viewed using standard image viewing software.

Cylindrical panoramas depict a horizontal field of view that is 360° around but has vertical constraints. The limits of the vertical field of view depend on the equipment used and/or the way the image is cropped. If flattened out, horizontal lines that are straight in reality become curved, while straight vertical lines remain straight.

A cylindrical panorama is intended to be viewed as if it were wrapped into the shape of a cylinder and viewed from within, and is stored, as a single .mov file, a single flattened image (with distortion), or as a series of rectilinear tile images within a single .mov file. Special software that can display a wrapped image, such as Apple's QuickTime Player is used, so as to avoid any unnatural curving or distortion.

Virtual Reality Images utilize spherical projection [BPG07], where a spherical panoramic image shows the entire field of view from a single point, 360° horizontally and 180° vertically, allowing the viewer to look in every direction, including the zenith and nadir. The image is wrapped into a sphere and viewed from the center, is distorted in the horizontal direction, and also slightly distorted in the

vertical direction, particularly at what would be the top and bottom "poles" of the sphere. It can be saved as a single .mov file and viewed with an application such as QuickTime, allowing the viewer to see the image without any unnatural distortion.

Pixel Selection and Weighting: Finally in order to create an attractive looking panorama, source pixels that have been mapped onto an appropriate surface, must be blended together. If all of the images are in perfect registration and identically exposed, combining and blending these pixels together, is completely straightforward. However, for real images, visible seams (due to exposure differences), blurring (due to mis-registration), or ghosting (due to moving objects) can occur [RS06], and the desired approach is to decide, which pixels to use and how to weight or blend them.

The simplest way to create a final composite is to take an average value at each pixel. Simple averaging usually does not work very well, since exposure differences, mis-registrations, and scene movement are all very visible. If rapidly moving objects are the only problem, taking a median filter (which is a kind of pixel selection operator) can often be used to remove them.

A better approach to averaging is to weight pixels near the center of the image more heavily and to down-weight pixels near the edges. When an image has some cutout regions, down-weighting pixels near the edges of both cutouts and edges is preferable. This can be done by computing a distance map or grassfire transform.

We call weighted averaging with a distance map ***"feathering"*** [42,199,210] and it does a reasonable job of blending over exposure differences. However, blurring, and ghosting can still be problems. One way to improve feathering is to raise the distance map values to some large power. The weighted averages then become dominated by the larger values, and the resulting composite can often provide a reasonable tradeoff between visible exposure differences and blur.

Optimal Seam Selection: Placing the seams in regions where the images agree, so that transitions from one source to another are not visible, is a very straightforward method to select the seams between regions where different images contribute to the final

[19]

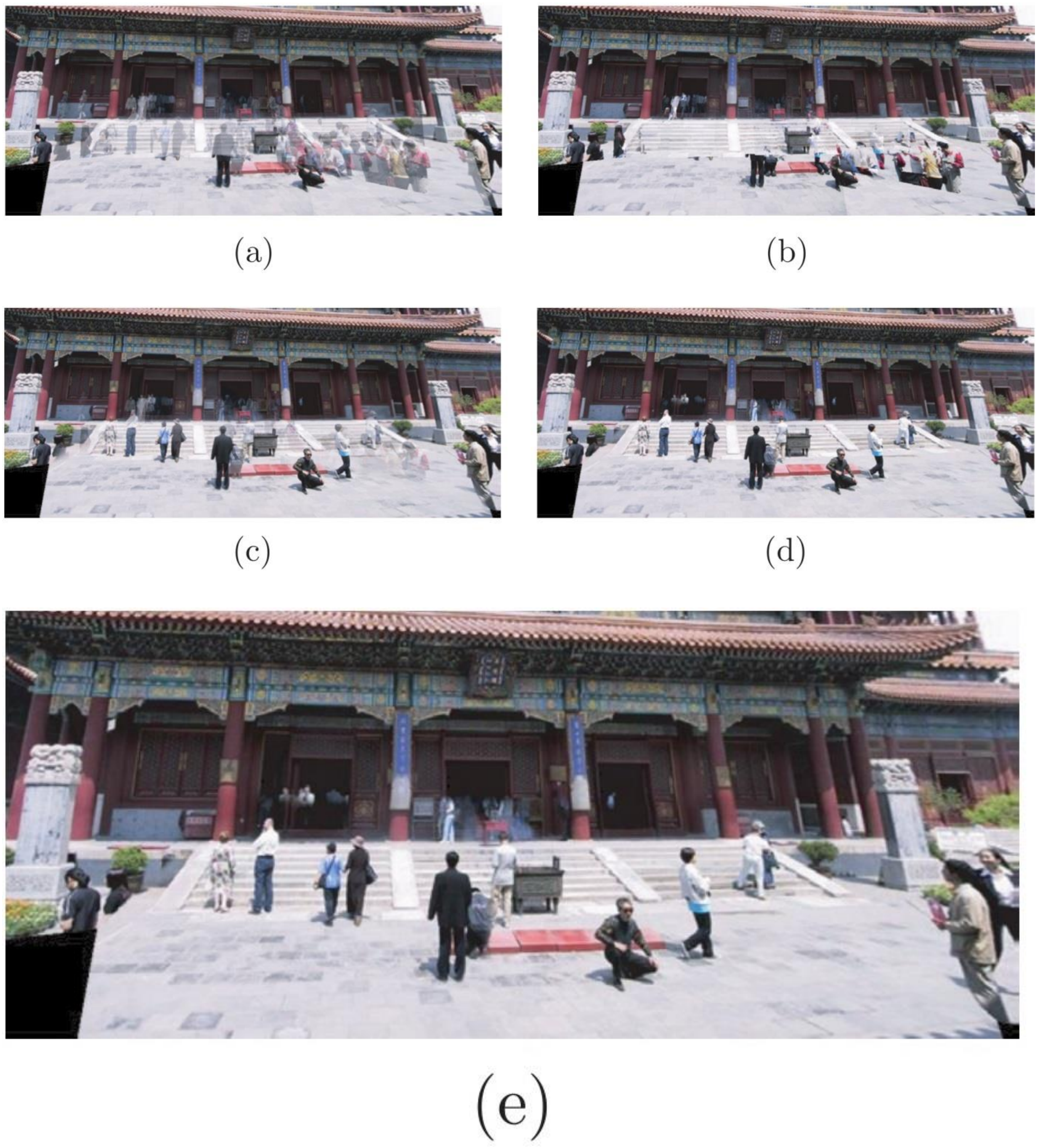


Fig. [19]
Blending images with different methods. (a) uses a simple averaging and (b) applies a median filter. (c) Weighted averaging weight pixels near the center of the image more heavily and to down-weight pixels near the edges (feathering). (d) One way to improve feathering is to raise the values to some large power, and the weighted averages become dominated by larger values (p-norm weighting). (e) Assigning each pixel to the nearest image center (Vornoi diagram).

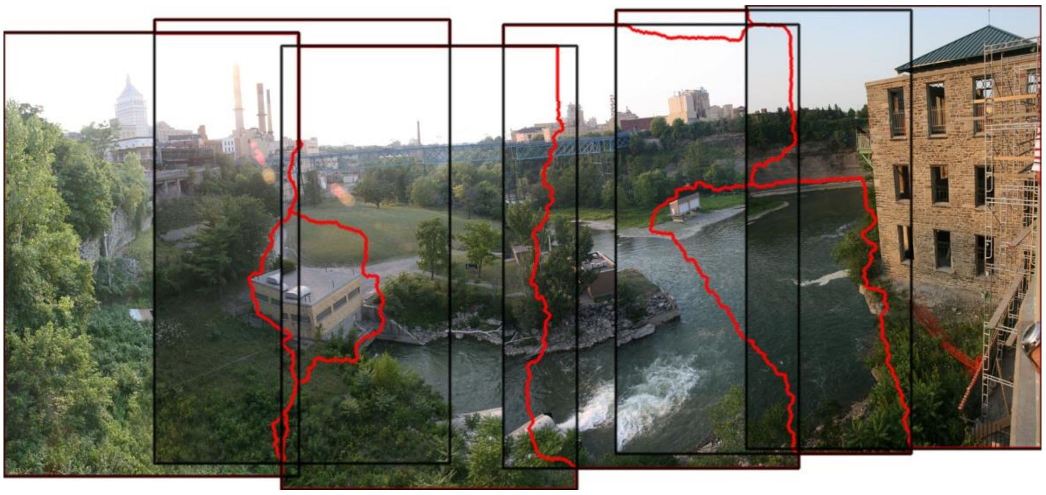


Fig. [20]
Registration, alligment and blending of multiple images. Red color denotes the seams between the images.

composite. The algorithm avoids “cutting through” moving objects where a seam would look unnatural. Moving objects produce the most visible artifacts, namely translucent looking ghosts, so we use algorithms to remove them.

Blending & Exposure Compensation: The final step, is to compensate for exposure differences and other mis-alignments. A novel approach to exposure compensation is to estimate a single high dynamic range (HDR) radiance map [DMB97], from the differently exposed images.

6. Real Life Experiments of Digital Image Stitching

i. *Panorama Tools and Hugin Overview*

The theory and the techniques, mentioned in the previous section, are implemented in **Panorama Tools** [PT13] and **Hugin** [HD19]. Panorama Tools (also known as PanoTools) are a suite of programs and libraries originally written by the German physics and mathematics professor Helmut Dersch. They create a powerful framework for re-projecting and blending multiple source images into immersive panoramas of many types. An updated version of the Panorama Tools library serves as the underlying core engine for many software panorama GUI front-ends [PT13].

Hugin is a cross-platform open source GUI front-end for Panorama Tools. It supports photo stitching and HDR merging and is developed by Pablo d'Angelo and others. Digital Stitching is based on the workflow described in section 5. Several overlapping photos are taken from the same location, control points are generated in all of them, and based on these control points, these images are then aligned and transformed, in order to be blended together in a larger image. The latest version of Hugin was released in 2019 [HD19].

Hugin is designed to be user friendly and straight forward. Control points generation, is an easy and optionally an automatic process, and along with image transforms optimizations, they are previewed in the same window so the user can see whether results are acceptable or not. On acceptable results, panoramas can then be fully stitched, transformed and saved in a standard image format. Three types of interface are offered (Simple, Advanced, and Expert), appealing to different needs and different expertise. Unfortunately, there seems to be no native support, for non-Latin characters, generating errors, when a non-Latin alphabet is used.

ii. *Experimentation: Real Life Image Capturing*

The purpose of the experiments, was to demonstrate the usability and user friendliness of the suit of libraries controlled by Hugin and used in creating digital panoramas. Since all of the algorithms and the workflow described above, are seamlessly integrated into libraries, acceptable results are achieved with minimal effort and input from users. Testing the conditions, that these algorithms can compensate for errors and make corrections automatically , is also examined.

We build the following examples using only the Simple and Advanced interfaces, since they are most widely used by novice and less experienced users. We tested different capture equipment and lighting conditions, and on one occasion a different suit of programs, other than Hugin.

(1) Establishing a Baseline: The Castle of Chora And Livadi Bay in Astypalaia Island Cases

Astypalaia Island is located near the center of Aegean, connecting the geographic regions of Cyclades and Dodecanese. The Castle of Chora was built in the 13th century, by Venetians that occupied the island at the time. It is not really a traditional castle, but it consists of a series of heavily built mansions, the type that the Normans used to call "keeps", so its architectural form is based in horizontal and vertical straight lines, making an ideal candidate for creating a panorama that can be used as reference, since the algorithms can easily extract common features between the images.

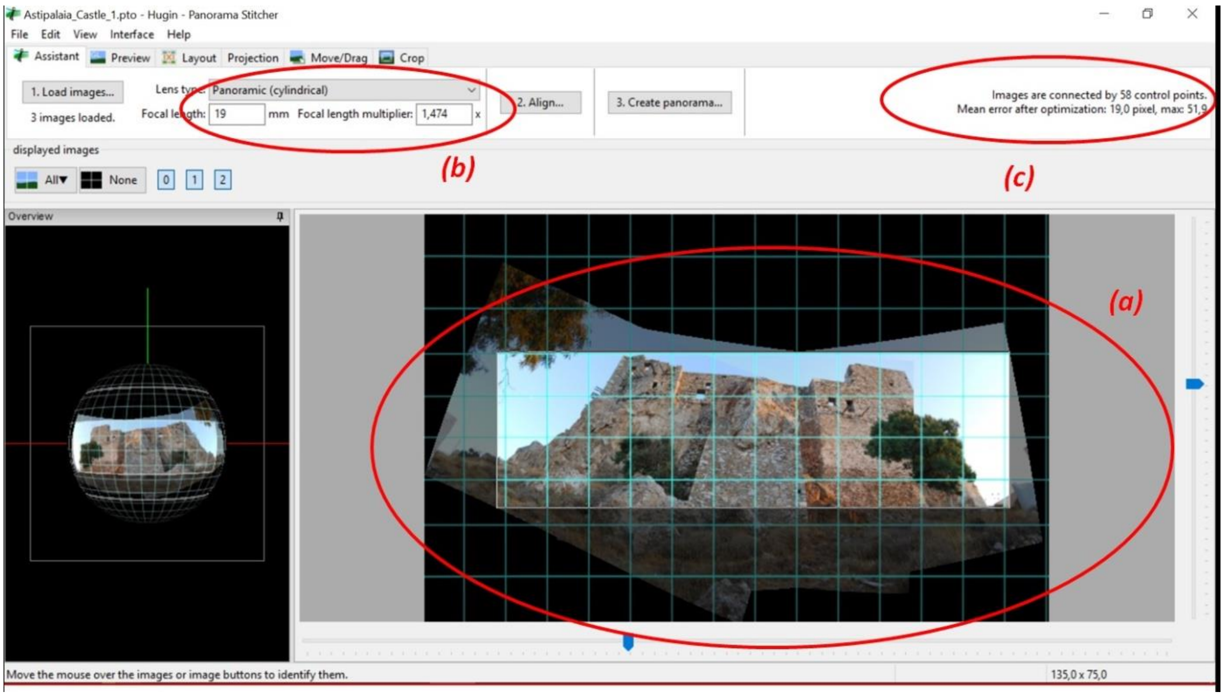
Three images of the western face of the castle, were captured with a Nikon D60 dSLR camera, with this camera pointing upwards from the ground, and the lighting conditions of an early afternoon, with no additional light sources and nor any tripod to stabilize it. Importing these images in Hugin, using the simple type of interface was straightforward and easy. After the registration and alignment phase, 58 control points were generated by the program and then used to connect the images. These control points could then be inspected and fined tuned if that was necessary in the preview window.

Before the registration and alignment process could begin, Hugin needed to extract information from the images, (stored as Exchangeable Image File Format – EXIF meta data in the image file, and generated by the camera at the time of image capture), regarding the type of lens and the Horizontal Field of View – HFoV used. From these values Hugin calculated the focal length, and the focal multiplier parameters needed, otherwise these values had to be manually entered.

After the registration and alignment processes were completed, different projections were applied to the previewing image, in order to avoid artifacts and distorted lines. Both rectilinear and cylindrical projections performed equally well, owning it to the architecture of the building with no horizontal lines prominent in the image. In fact even after rotating left and right images and while keeping the central image as a reference, no visible errors were detected. A cylindrical projection was finally used, since it provided slightly better aesthetically results, especially after cropping the image.

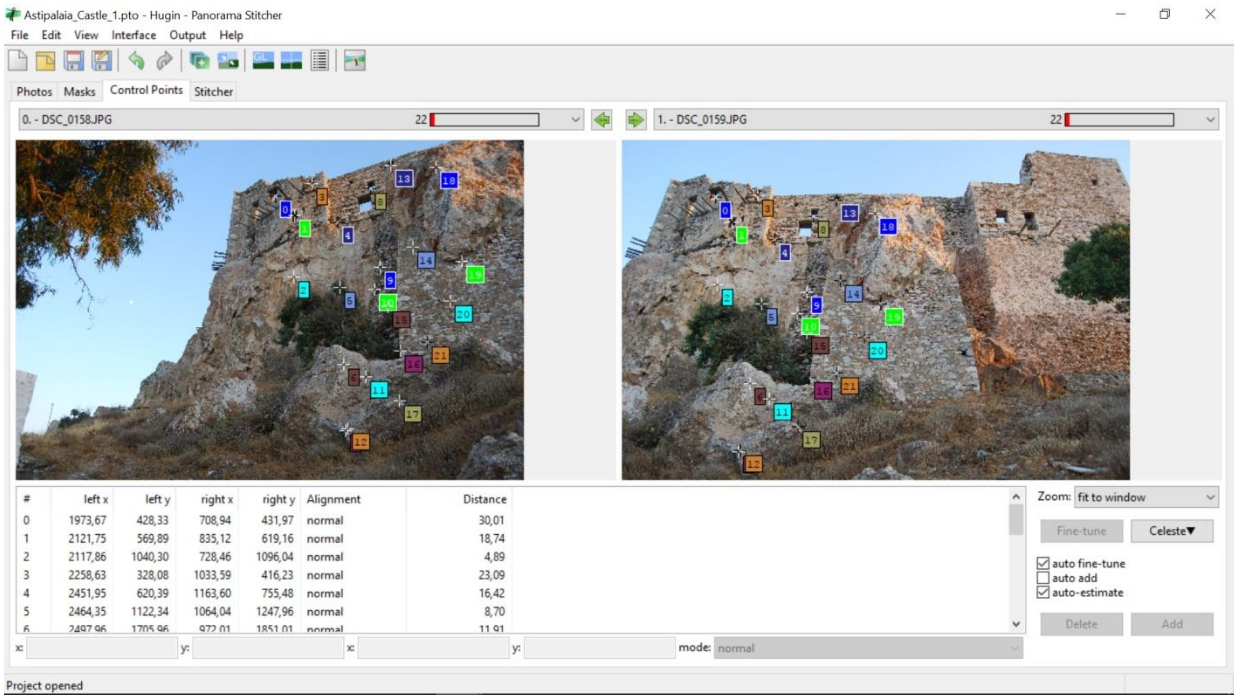
Switching to Hugin's advanced interface (Fig. 22), we can observe the distribution of control points. These control points are used to align and stitch the images together and are the final products of feature detection and feature matching methods described earlier. When we applied a simple pass of the SURF algorithm to the images, (Fig. 23 - Open SURF implementation in MatLab (SURF_Detection_Matching.m)), the general distribution of control points generated by Hugin, coincided with common features discovered by SURF. Hugin refines and fine tunes these common features, and transforms them into usable control points.

Livadi Bay is located, on the eastern part of Astypalaia island, and is presented in a series of images captured from the eastern cliff of the bay facing west. Testing Hugin's capabilities in combining a


Astipalaia_Castle_1.pto - Hugin - Panorama Stitcher
File Edit View Interface Help
Assistant Preview Layout Projection Move/Drag Crop
1. Load images...
2. Align...
3. Create panorama...
displayed images
All▼ None 0 1 2
Overview
(a)
(b)
(c)

3 images loaded.
Lens type: Panoramic (cylindrical)
Focal length: 19 mm Focal length multiplier: 1,474 x
Images are connected by 58 control points.
Mean error after optimization: 19,0 pixel, max: 51,9
Move the mouse over the images or image buttons to identify them.
135,0 x 75,0

Fig . [21]
Hugin Simple Interface. This part of the interface is designed to be easy to use and straightforward. Castle of Chora series of images, are loaded in order to be processed. (a) Panorama preview window. Left and right images are rotated in order to be aligned with the central image. (b) Lens type and characteristics. These values are usually extracted from EXIF metadata. (c) Generated control points. They are created by the use of feature detection matching algorithms, in the three images.


Astipalaia_Castle_1.pto - Hugin - Panorama Stitcher
File Edit View Interface Output Help
Photos Masks Control Points Stitcher
Fine-tune Celeste▼
Delete Add

0. - DSC_0158.JPG 22
1. - DSC_0159.JPG 22

#	left x	left y	right x	right y	Alignment	Distance
0	1973,67	428,33	708,94	431,97	normal	30,01
1	2121,75	569,89	835,12	619,16	normal	18,74
2	2117,86	1040,30	728,46	1096,04	normal	4,89
3	2258,63	328,08	1033,59	416,23	normal	23,09
4	2451,95	620,39	1163,60	755,48	normal	16,42
5	2464,35	1122,34	1064,04	1247,96	normal	8,70
6	2497,96	1705,96	972,01	1851,01	normal	11,91

Zoom: fit to window
☑ auto fine-tune
☐ auto add
☑ auto-estimate
mode: normal
Project opened

Fig. [22] Hugin is switced to its advance type of interface. Left and center images, that will be joined to compose the digital panorama, are loaded. The numbered collored rectangles are control points that were generated after proccessing common features between these images. Even if this procedure is automatic, there are tools to further refine their positioning.

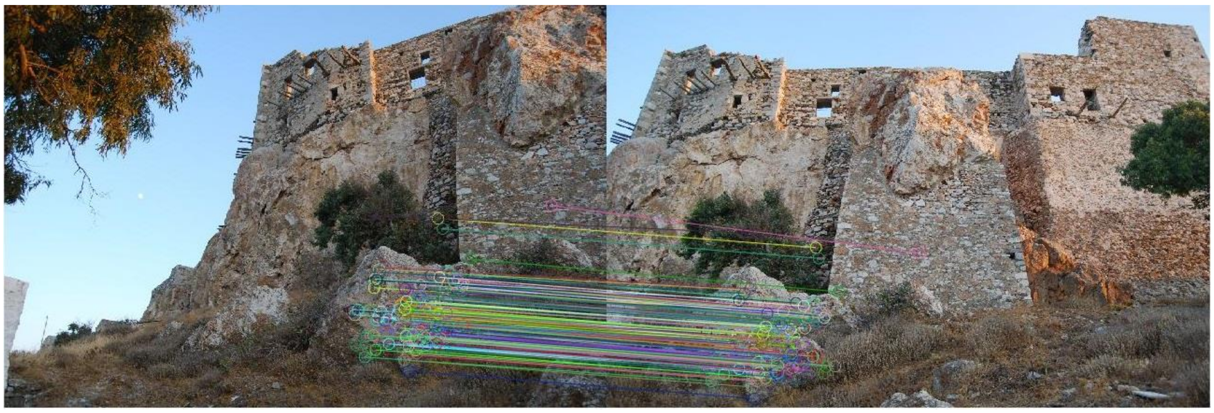


Fig. [23]
Looking for common features between the above images, using a MatLab SURF implementation (SURF_Detection_Matching.m). Some of the detected common features, after several processing iterations will evolve to control points generated used by Hugin.

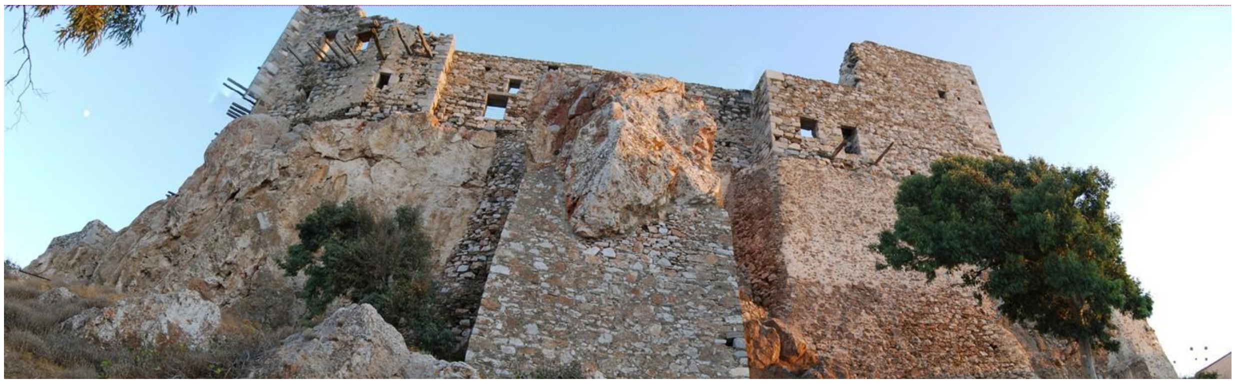
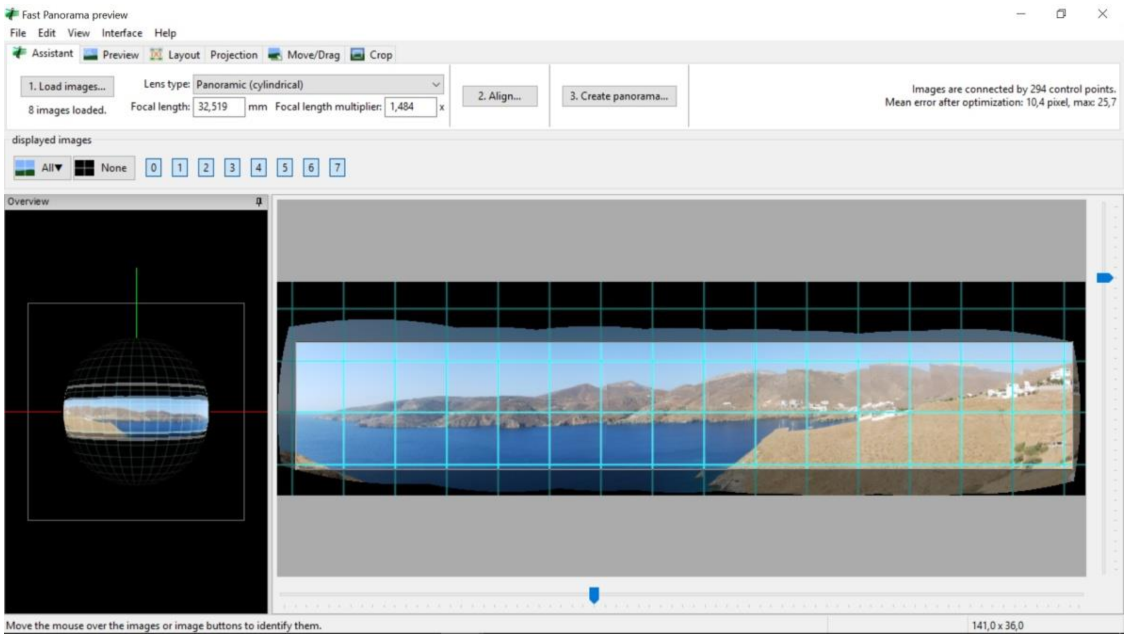


Fig. [24]
A new digital composite (panorama). The castle itself, is a series of mansions forming a defence structure.

greater number of images in a single composite with minimal effort from user, eight overlapping images that were shot in sequence, with the same Nikon D60 dSLR, this time mounted on a tripod in order to avoid unnecessary distortions and parallax errors (Fig. 25).

Images were then imported in Hugin, were aligned and combined into a panorama composite. Different projections were also applied in the preview window, with rectilinear and cylindrical projections both performing equally well. Since there were no prominent horizontal lines to create distortions, and since the final composite looked slightly better with a cylindrical projection, this projection was used. For the final eight image composite, a total of 294 control points were generated. The west face of the bay, at high noon as seen from the eastern shore, is depicted (Fig. 26).

We finally then established Hugin's capabilities, of combining many images into a single composite, in a small amount of time and with minimal input, with the creation of the composites above. The only requisitives lied, in the equipment used (a dSLR with or without a



8 images loaded.

Lens type: Panoramic (cylindrical)

Focal length: 32,519 mm Focal length multiplier: 1,484 x

Images are connected by 294 control points.
Mean error after optimization: 10,4 pixel, max: 25,7

Move the mouse over the images or image buttons to identify them.

141,0 x 36,0

Fig . [25]
This time eight overlapping images were shot in sequence, using a dSLR and a mounting tripod. They were loaded in Hugin (simple interface pictured) in order to create a panorama of the Livadi Bay. Since the camera was mounted to the tripod, no distortions or parallax errors were noticed.

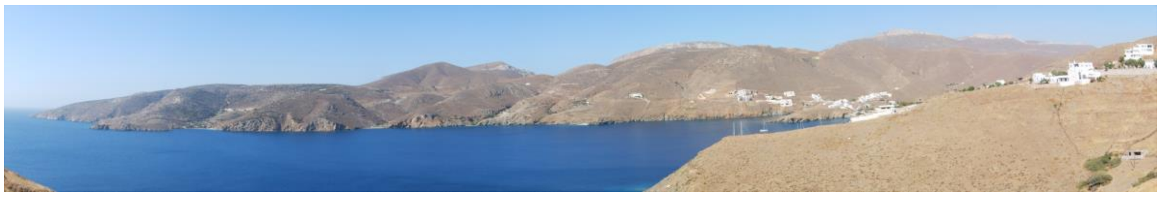


Fig. [26]
Livadi Bay is located on the eastern part of Astypalaia Island. This is the west shore of the bay.

tripod), and an overlap needed between successive images, at the time of capture (a 10% overlap is sufficient in most cases).

(2) Parallax, Lighting and Projection Errors: Kilindra Inlet, Koumoundourou Lake and Portaria, Pilio

Kilindra Inlet lies in the eastern shore of Livadi Bay (Fig. 27). When these images were captured, it was late morning with corresponding lighting conditions. Five overlapping images, shot in sequence, were taken with the same dSLR used above. Capturing these images over a surface covered by the sea, with different reflecting qualities presented over a large percentage of surface, was considered to be a very good example of testing the capabilities of Hugin and Pano tools.

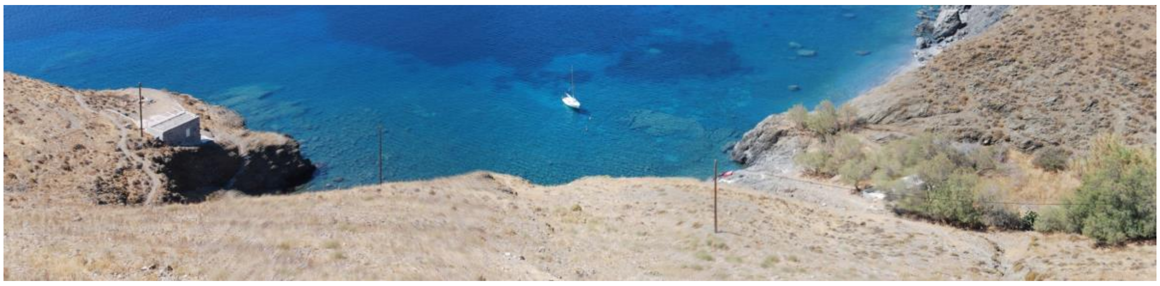


Fig. [27]
Kilindra Inlet, lies on the eastern shore of Livadi Bay. It was late morning at the time of capture, with the sea at the foreground. Different shades of blue, represent different reflecting qualities of the sea water.

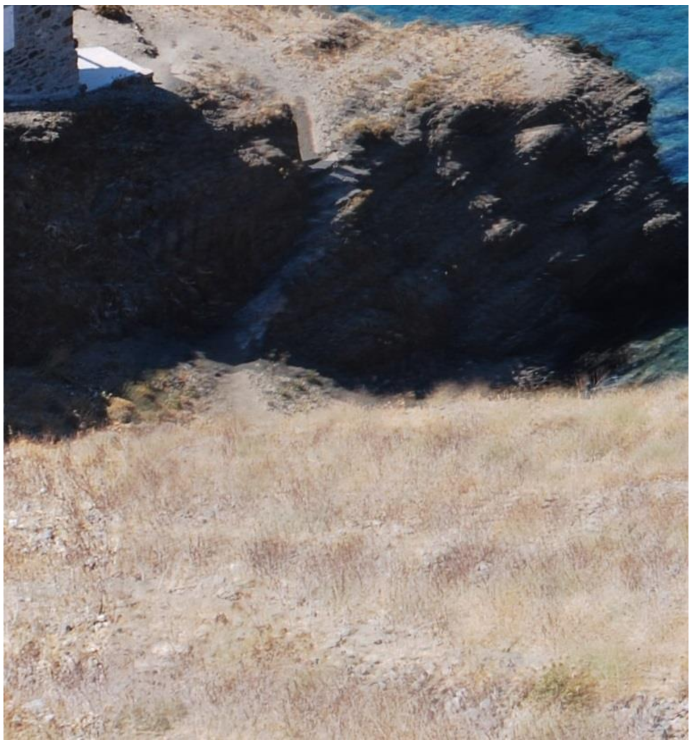


Fig. [28]
The part of the image that is blurred, due to a parallax error. It seems that the camera was unconsiously moved, at the time of capture, pointing at the significance of using mounting equipment, when trying to create complex scenes.

The camera was not mounted on a tripod this time. The wide area covered by sea water, with different shades of blue presented in each successive image, were successfully stitched in a wider composite image. These different areas presented well defined "blobs" that the algorithm was able to exploit. On the other hand a small parallax error crept into the image, identifiable as a blur in a small part of it. (Fig. 28).

The algorithm was not able to compensate for the error, at this particular area of the image. Possibly it could not detect or generate usable control points since the area depicted is quite coarse, with no discerning changes. But this error is not attributed to Hugin, but to human operation during the time of capture.

In Koumoundourou Lake image, we used a different procedure and a different camera. With an iPhone v5 camera, instead of a dSLR, we captured these images in one continuous take, without any stabilizing equipment, and the included mobile app created the composite (Fig. 29). Capturing the images, did not presented any problem, but in creating the panorama composite, a cylindrical projection was used by default, resulting in straight horizontal lines (roads and other surfaces) appearing as curves. These artifacts are more pronounced in the roads depicted in the image (Fig. 30). After we compared the image of the same road in Google Earth with the composite, it was clear post processing of the images curved the road, since in reality this particular road forms a nearly straight line (Fig. 31).

Portaria, Pilio series of images, is an evolution of the previous procedure. We also captured these images with an iPhone v5s mobile phone camera. However, instead of using a mobile app, we imported these images into Hugin in order to construct a cylindrical projection composite. Extra care was taken to create a stable surface to support the camera, in order to avoid parallax errors. But the results were not satisfactory since images were misaligned, and this is more prevalent in the background.

We re-iterated the procedure, with a rectilinear projection, and we constructed a new version of the same composite. There are still some misalignments in the background, but this time there are too small to notice, and are attributed to the lack of a sophisticated zoom function in the mobile camera, especially when images of far horizon objects are captured (Fig. 32).

(3) Image misalignment: Demetrius Shipwreck - Githio

The objective of Agios Demetrius, Githio series of images, was to determine if it was possible to create a composite, with images captured from a video source. A suitable scene needed the camera to travel in nearly straight line without significantly changing the perspective. After examining many videos uploaded to Internet, scenes

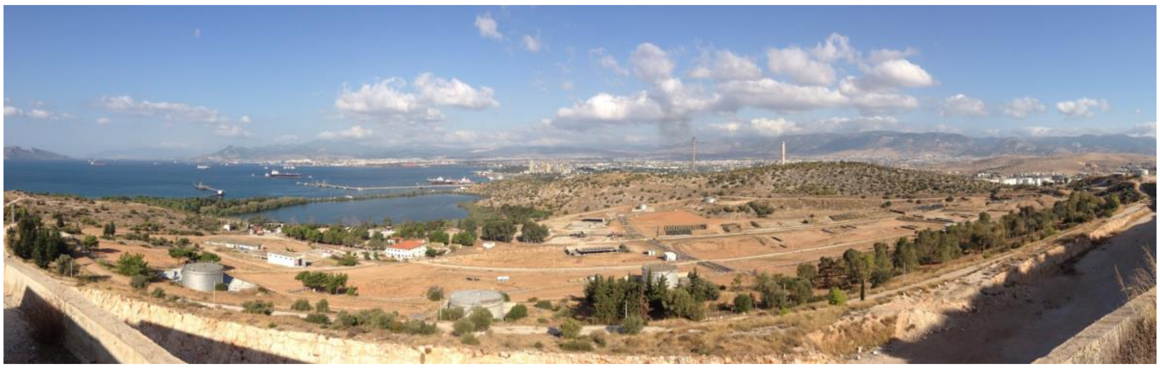


Fig. [29]
This composite was created with an iPhone v5, in a single take without combining different images. Image post processing, was also done on the phone.

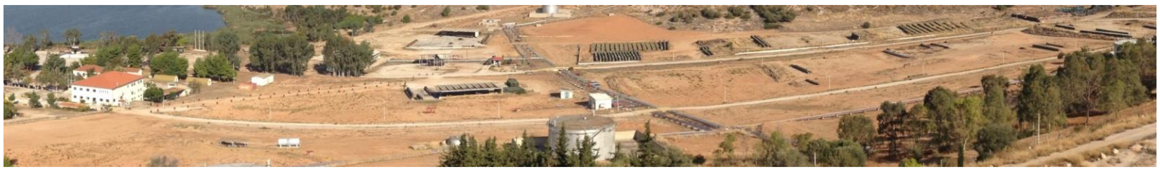


Fig. [30]
It seems that straight line image objects such as this road, are appearing curved. This points to incorrect image projection (cylindrical instead of planar/rectilinear)

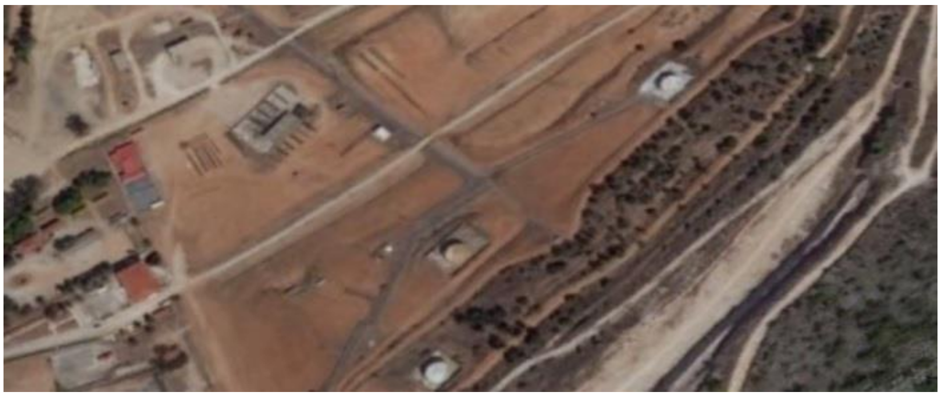


Fig. [31]
Same road as seen from Google Earth. It is straight and does not make a bend.

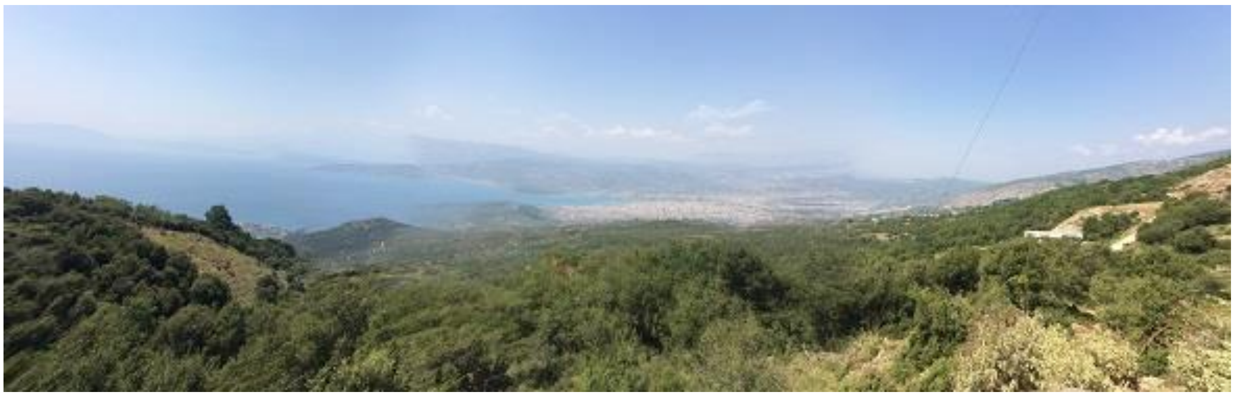


Fig. [32]
These images were captured by a mobile phone, but were processed with Hugin. There are some misalignments in the background, that are attributed to the lack of a sophisticated zoom function in the mobile camera, especially when images of far horizon objects are captured. Similar images captured with dSLR, produced far better results.

from a promotional 4k video created by "Drone Solutions" [DS16] depicting the shipwreck of "Demetrius", a cargo ship that run aground in Githio bay in the '80s, were deemed suitable.

We imported the video file in VLC media player [VL19], and we captured screenshots, as .jpg images, with a capture - sampling rate of 1/10. (One out of ten images). After a close examination, out of nearly 200 images, we selected two of them and then we imported them into Hugin (Fig. 33 a & b). The first problem arose, when Hugin was not able to find and import any EXIF metadata, and asked for FoV information.

We set the value of FoV to 50mm, a value commonly used in commercial equipment, and both images were finally loaded (Fig. 34). The second problem occurred in the alignment phase, since a lot of errors were generated, and the images were not correctly aligned in the preview window. Continuing with the procedure, the produced composite featured a very strong parallax effect (Fig. 35).

With unsatisfactory results and a final composite that featured a lot of ghosting artifacts, we switched Hugin to advanced interface mode, and we re-iterated the procedure started from the beginning, but this time we introduced our control points manually, after we carefully examined the images. Hugin normally generates 20 to 30 control points to accurately align a series of images, in a time period counted in seconds. Selecting, positioning and finally fine-tuning just six control points, it took us nearly two hours (Fig. 36). Even though this time, we eliminated most of the ghosting artifacts, images were still misaligned, although this time this was not a very pronounced effect (Fig. 37).

In order to map common features between these two images, and get an idea of how Hugin utilised them, we loaded both of them into Matlab, and implemented the SURF algorithm. (OpenSURF library implementation in the "SURF_Detection_Matching.m" script). With this script, we can detect and reveal a desired number of common features between two images, and we unveiled a very large number of them, after executing it. (We set the value in the script to 450).

The distribution of these common features was concentrated in two parts of the images (Fig. 38). Some of them coincided, with both the manually inserted control points, and with the control points that Hugin generated. SURF algorithm worked as designed and detected "blob structures, especially in those features that were

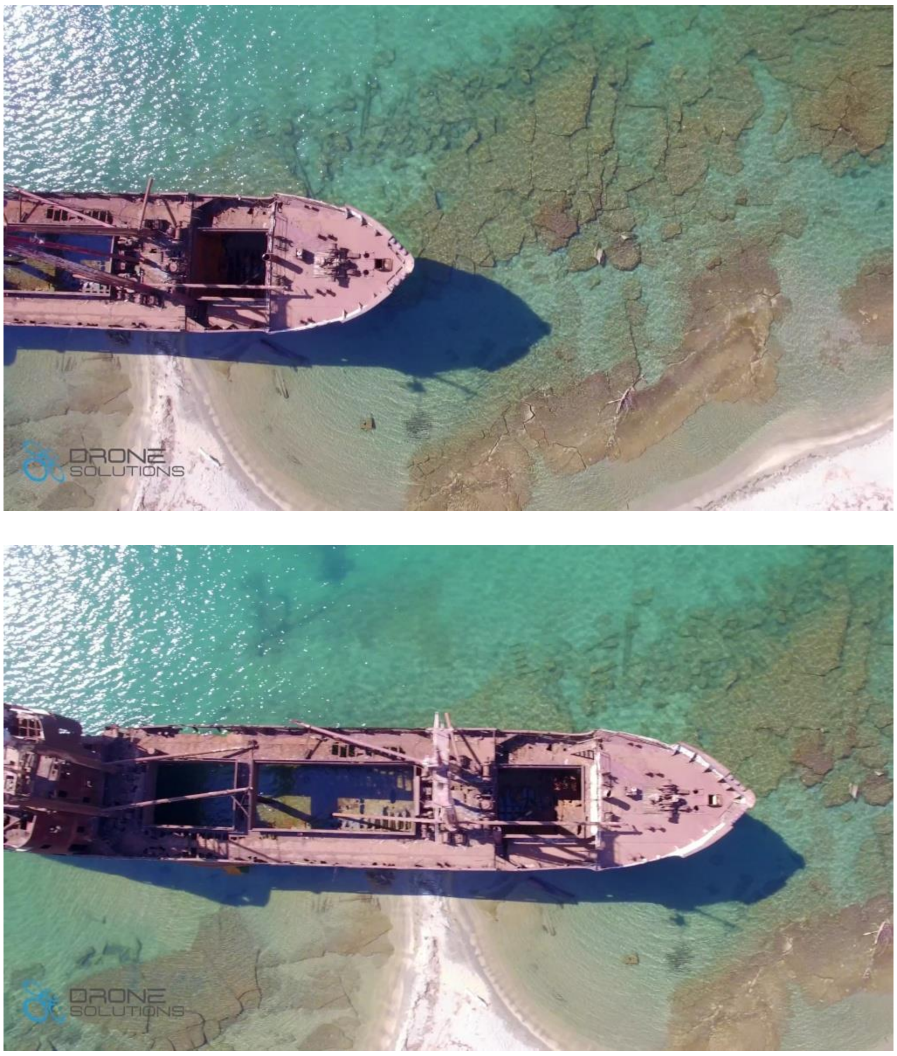
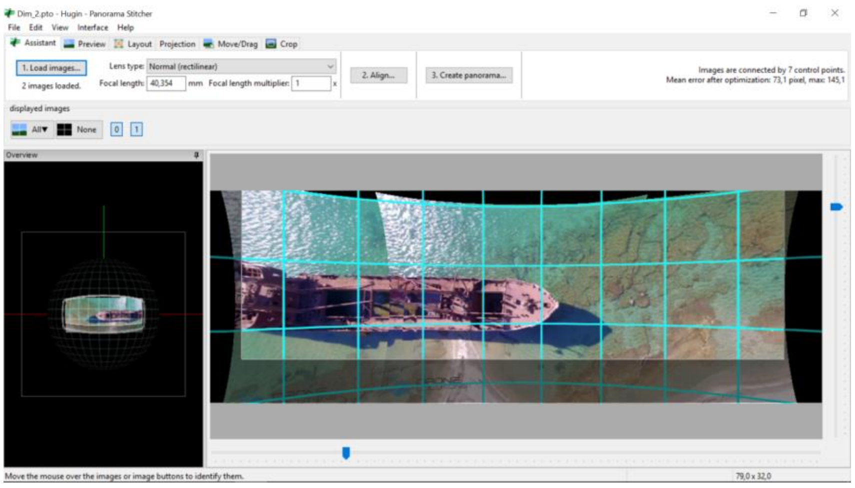


Fig. [33 a & b]
These two images (out of nearly 200 extracted from video), were suited to create a composite. There are many common features between them, and much overlapping so it is possible that many suitable control points will be generated.



2 images loaded.
Lens type: Normal (rectilinear)
Focal length: 40,354 mm Focal length multiplier: 1 x
Images are connected by 7 control points.
Mean error after optimization: 73,1 pixel, max: 145,1
Move the mouse over the images or image buttons to identify them.
79,0 x 32,0

Fig. [34]
Preview window of Hugin's simple intereface. Since no EXIF metadata were found, FoV was set to a common value of 50.

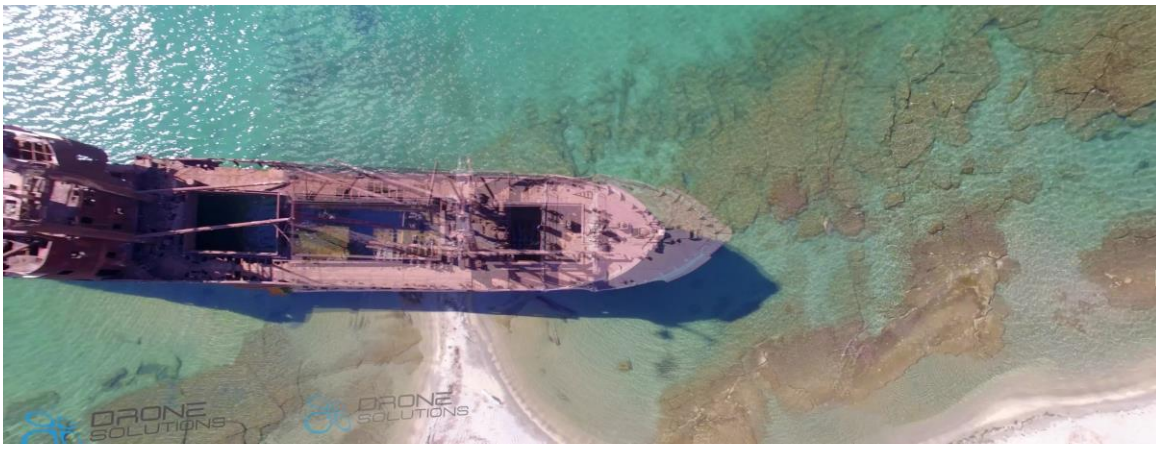
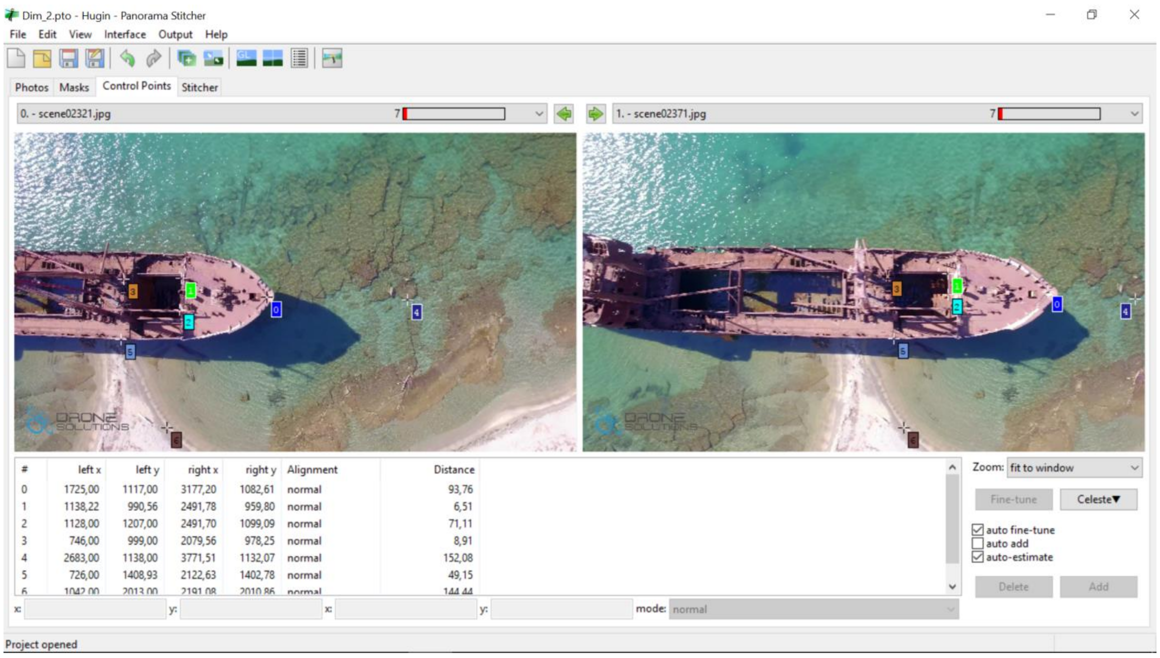


Fig. [35]

A very strong ghosting (parallax) effect. Ghosting effect occurs when, an object moves between captures, but in this case it seems that the camera changed slightly its perspective.



#	left x	left y	right x	right y	Alignment	Distance
0	1725,00	1117,00	3177,20	1082,61	normal	93,76
1	1138,22	990,56	2491,78	959,80	normal	6,51
2	1128,00	1207,00	2491,70	1099,09	normal	71,11
3	746,00	999,00	2079,56	978,25	normal	8,91
4	2683,00	1138,00	3771,51	1132,07	normal	152,08
5	726,00	1408,93	2122,63	1402,78	normal	49,15
6	1042,00	2013,00	2191,08	2010,86	normal	144,44

Zoom: fit to window

[✓] auto fine-tune
[ ] auto add
[✓] auto-estimate

mode: normal

Project opened

Fig. [36]

Hugin's advanced intereface. Manually introducing six control points. Hugin normally generates 20 to 30 control points to accurately align a series of images, in a time period counted in seconds. Manually introducing these six control points, took nearly two hours.

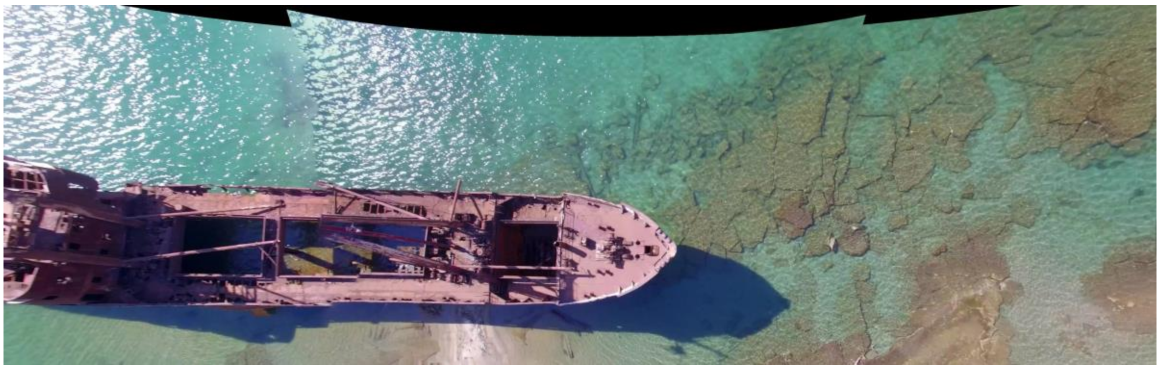


Fig. [37]

No parallax errors but the images show a very small misalligment.

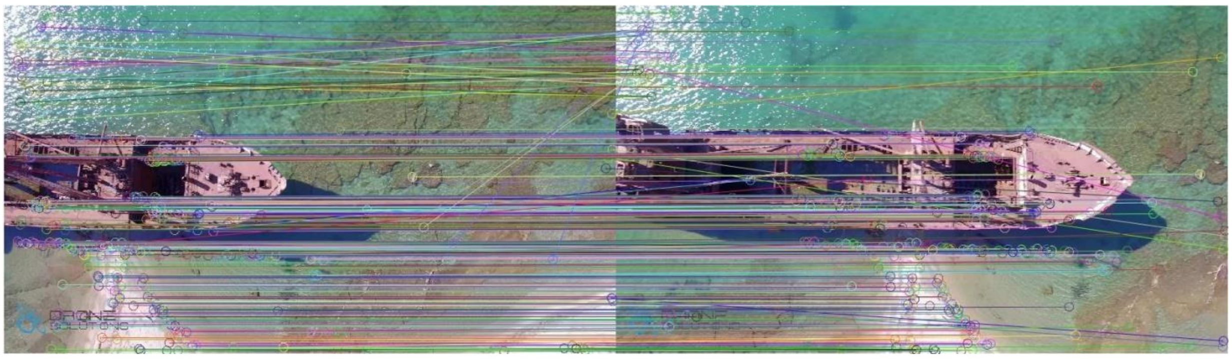


Fig. [38]
We implementing the SURF algorithm in both images. We discovered nearly 450 common features, in those areas of the images that had either greater light exposure or were resembled geometric forms, clearly forming "blob" structures.

located in areas of the images that had either greater light exposure, or were resembling geometric forms (glint in the images, or the shape of the beach). Hugin on the other hand fine-tuned this selection, and only concentrated on ten of those features, located parts of the images, that was common between them. (The nearly triangular shaped beach, in the lower part).

Having established that algorithms and procedures used by Hugin, for control points generation worked as intended, the presence of ghosting artifacts, was put under examination. After re-examining the original video, we found out that, in the course of its flight path, the platform changed its camera perspective slightly, and the algorithms could not compensate for it. Since in earlier examples (Fig. 28) we established that sudden changes in the motion of the camera could introduce ghosting artifacts, it is reasonable to attribute these errors to perspective changing, caused by the camera operation.

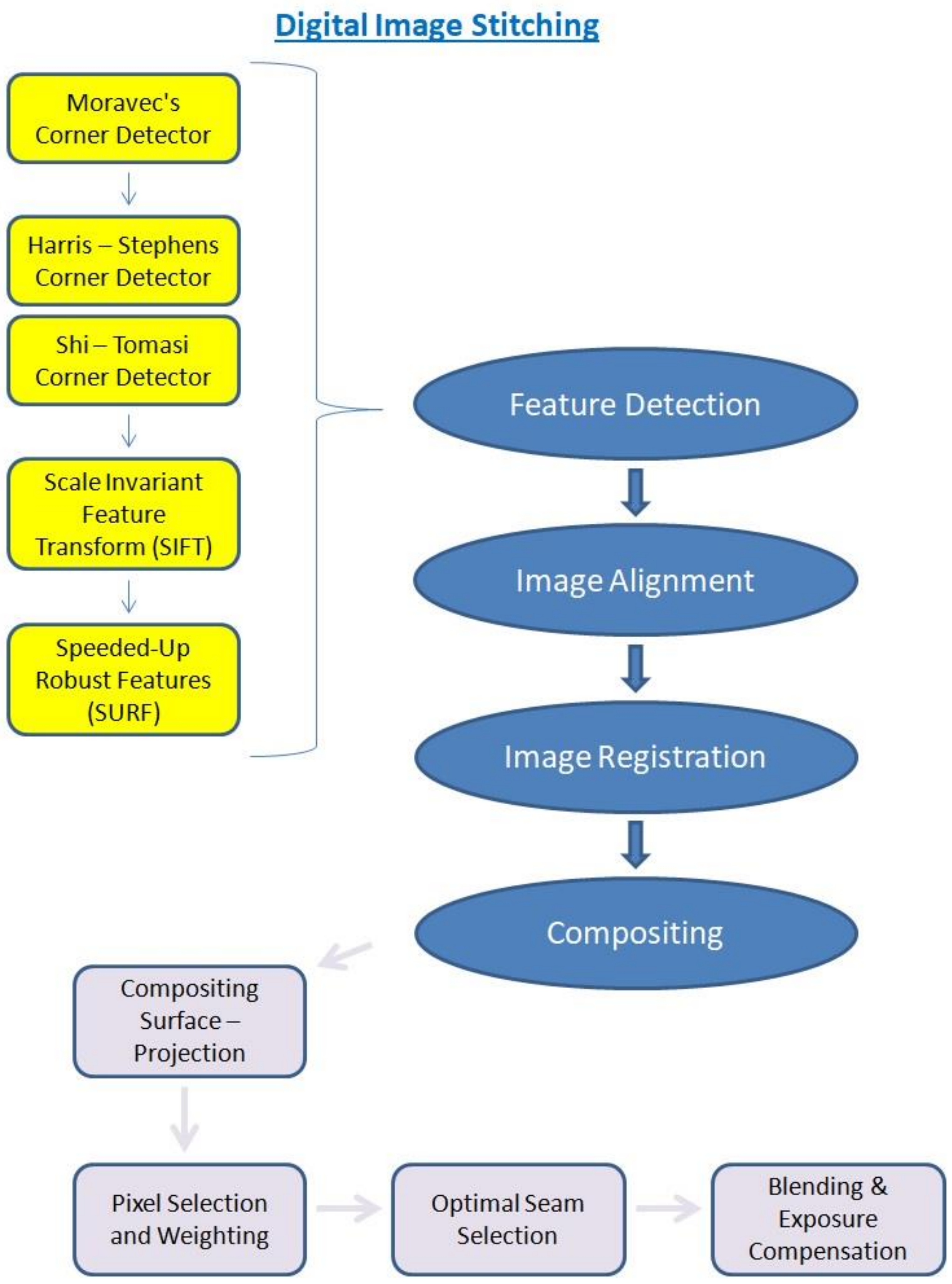
Summarizing, an experienced user of Hugin can create panorama composites from video captured images with acceptable results; however, it seems that Hugin is optimized to processing images captured with a commercial dSLR camera. Moreover, for an inexperienced user, the simple type of interface of Hugin creates composite panoramas far quicker, and less prone to errors than the ones we create, by manually introducing control points.

iii. *An Afterword on the experiments*

Hugin simplifies the process of creating digital composites - panoramas. As the above examples show, all we need is a common dSLR camera and a steady hand. Even though we caused most of the errors , from poor camera handling at the time of image capture, Hugin did a good job, in compensating for most of them.

One the other hand, if we are in the habit of using images from the web (still images or images captured from video), lack of EXIF metadata, or the unknown lighting conditions in time of capture can complicate the procedure immensely and we need more time and considerable more experience, in order to produce acceptable results.

All in all Hugin/Pano Tools suit of algorithms is a very effective and streamlined method of creating digital panoramas since it automates the very complex procedures shown at chapters 4&5.


Digital Image Stitching
Moravec's Corner Detector
Harris – Stephens Corner Detector
Shi – Tomasi Corner Detector
Scale Invariant Feature Transform (SIFT)
Speeded-Up Robust Features (SURF)
Feature Detection
Image Alignment
Image Registration
Compositing
Compositing Surface – Projection
Pixel Selection and Weighting
Optimal Seam Selection
Blending & Exposure Compensation

7. 4D Panoramas: Lightfields

The light-field is a vector function that describes the amount of light flowing in every direction through every point in space. First proposed by Michael Faraday [MF1846] he theorized, that light should be interpreted as a field, much like the magnetic fields on which he had been working for several years. The phrase light-field was coined by Andrey Gershun, in a classic paper on the radiometric properties of light in three-dimensional space [AG1939].

Light-field photography offers a new approach to digitally captured images. New types of cameras, now commercially available, are able to capture every distinct light ray on their lenses, creating a 4D light field contained in a single image. This allows for a variety of image processing capabilities that traditional cameras do not offer. For example, the image can be digitally refocused, after it is captured and its depth can be estimated [MDY15].

The key enabling insight of light-field imaging, is a reinterpretation of the classic photographic imaging procedure, that separates the process of imaging a scene (i.e., scene capture) from the actual realization of an image (i.e., image synthesis)— a reinterpretation that offers new flexibility in terms of post processing. The underlying idea is that capturing the image in a single take, can offer far more possibilities than simple image processing (Fig. 39). Modern cameras are powerful computers that enable the execution of sophisticated algorithms to produce high-quality two-dimensional (2D) images.

Lightfield imaging is, however, moving beyond that level, by purposefully modifying classical optical designs to enable the capture of high dimensional data sets (in 4D mainly), that contain rich scene information, imaging out-of-focus regions and also capturing the full 3D content of a scene. The 2D images presented to the human observer are processed versions of the higher-dimensional data the sensor has acquired and only the computer sees in their raw form [RMD16].

The Plenoptic function

The theoretical background for light field imaging is **the plenoptic function** [MDY15], which is a ray-optical concept that assigns a radiance value to rays propagating within a physical space. It considers the usual 3D space to be penetrated by light that propagates in all directions. In doing so, light can be blocked, attenuated, or scattered [RMD16].

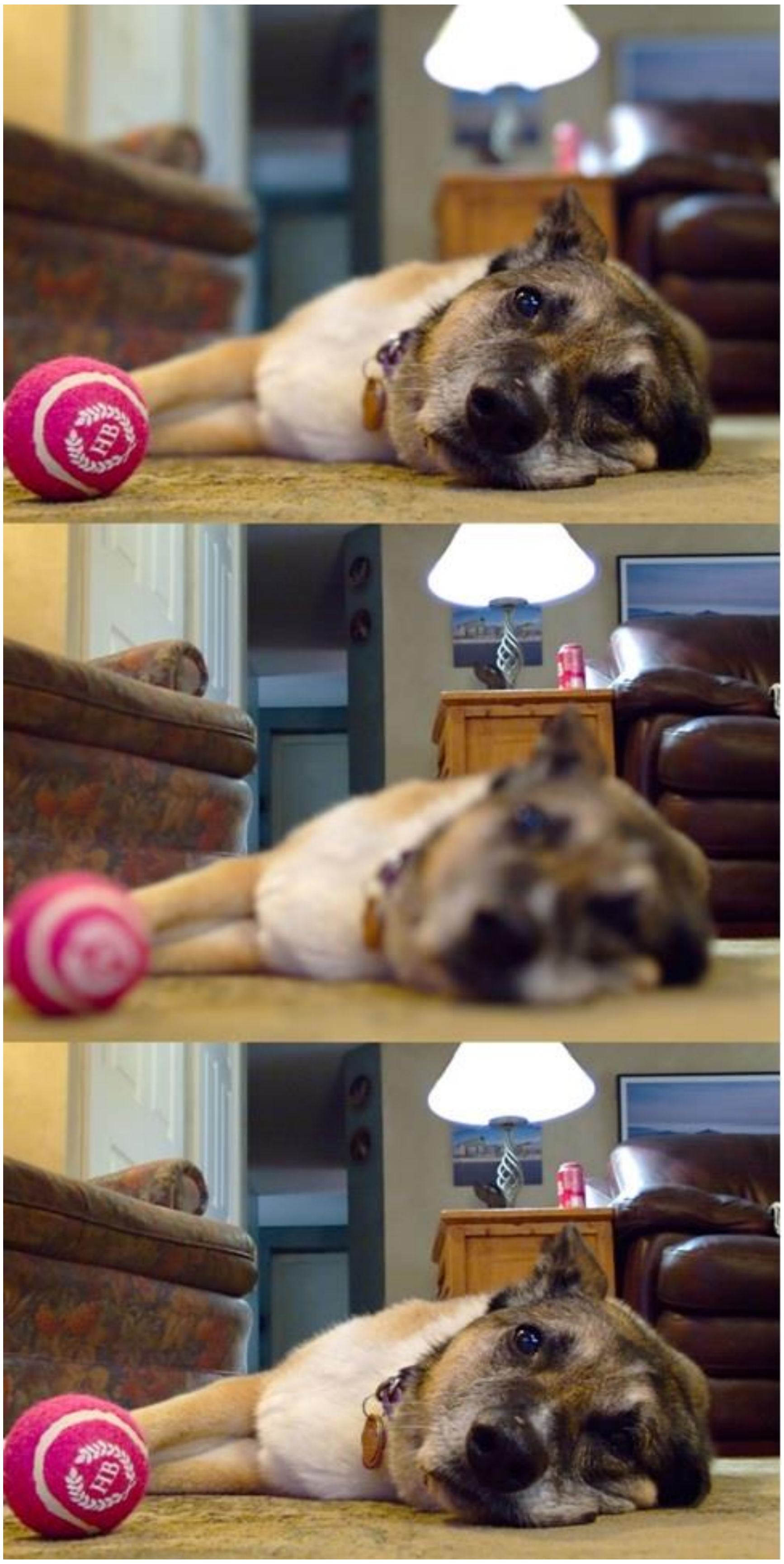


Fig. [39]
Light-field imaging, offers a new approach to digital photography. The focal distance and depth of field, can be altered in post processing after a photo is taken. - Near focus (top), Far focus (middle), Full depth of field (bottom).

However, instead of modeling this complexity as, e.g., computer graphics is doing, the plenoptic function is an unphysical, modeless, purely phenomenological description of the light distribution in the space. To accommodate for all the possible variations of light without referring to an underlying model, it adopts a high-dimensional description: arbitrary radiance values can be assigned at every position of space, for every possible propagation direction, for every wavelength, and for every point in time [RMD16].

To measure the plenoptic function one can imagine placing an idealized eye at every possible (Vx, Vy,Vz) location and recording the intensity of the light rays passing through the center of the pupil at every possible angle (θ , φ), for every wavelength, λ, at every time t. It is simplest to have the eye always look in the same direction, so that the angles (θ , φ) are always computed with respect to an optic axis that is parallel to the Vz axis. The resulting function takes the form:

$$P = P(\theta, \varphi, \lambda, t, Vx, Vy, Vz)$$

(Fig [40 - 42])

If we examine more closely, it seems that a large thread of image-based rendering work has been based on different dimensional expressions of the plenoptic function. The original 7D plenoptic function is defined as the intensity of light rays passing through the camera center at every location, at every possible viewing angle, for every wavelength and at any time. It has also been shown that light source directions can be incorporated into the plenoptic function for illumination control [WHON97].

By ignoring time and wavelength, a continuous 5D plenoptic function can be generated from a set of discrete samples [MB95], while if the scene or conversely the camera view can be constrained to fixed spatial dimensions, the plenoptic function is defined by 4 parameters (4D plenoptic function --> Plenoptic Cameras).

If finally the viewpoint is fixed and only the viewing directions and camera zoom can be altered, the plenoptic function simply becomes a 2D panorama (cylindrical [Che95] or spherical [SS97]).

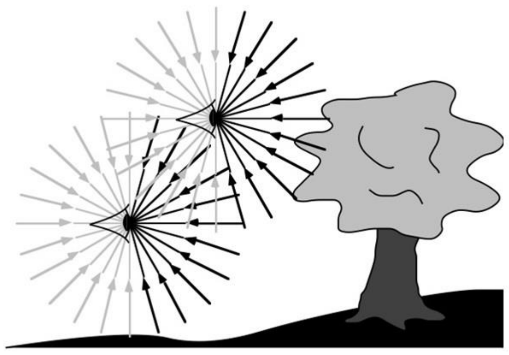
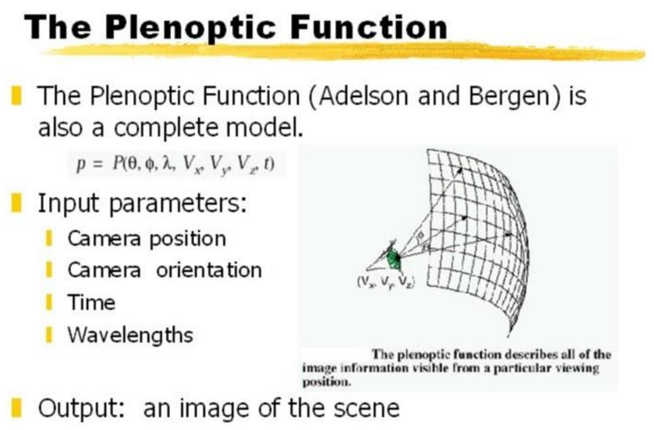


Fig.[40]
The plenoptic function describes the information available to an observer at any point in space and time. Shown here are two schematic eyes-which one should consider to have punctate pupils-gathering pencils of light rays. A real observer cannot see the light rays coming from behind, but the plenoptic function does include these rays.



The Plenoptic Function

- The Plenoptic Function (Adelson and Bergen) is also a complete model.

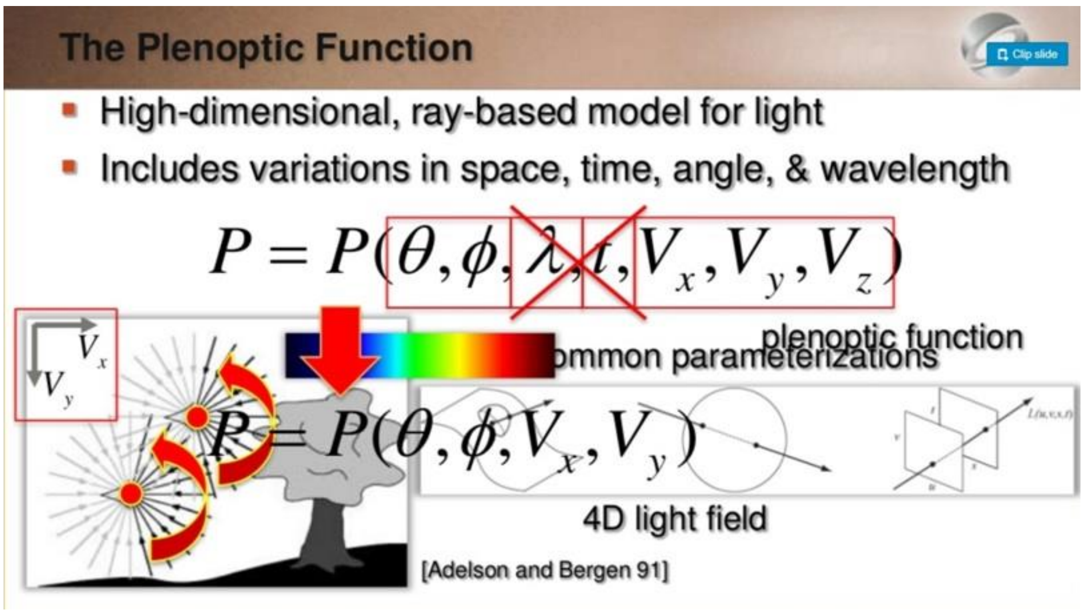
$p = P(\theta, \phi, \lambda, V_x, V_y, V_z, t)$

- Input parameters:
  - Camera position
  - Camera orientation
  - Time
  - Wavelengths

The plenoptic function describes all of the image information visible from a particular viewing position.

- Output: an image of the scene

[Fig. [41]](#)
The plenoptic function describes, all of the image information visible, from a particular viewing position.



The Plenoptic Function

- High-dimensional, ray-based model for light
- Includes variations in space, time, angle, & wavelength

$$P = P(\theta, \phi, \cancel{\lambda}, \cancel{t}, V_x, V_y, V_z)$$

plenoptic function

common parameterizations

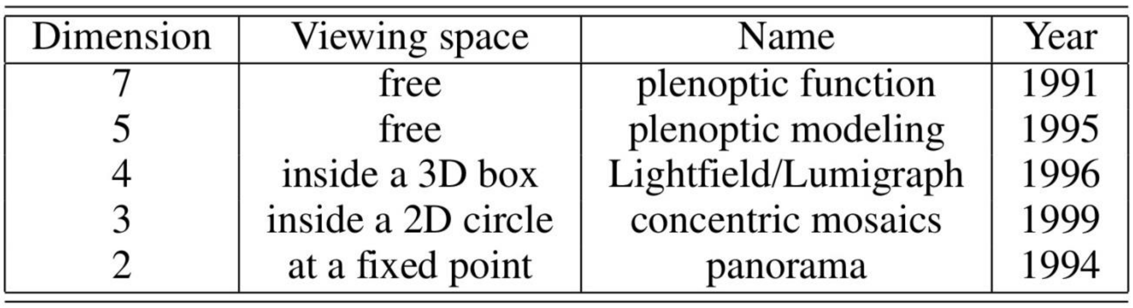
$$P = P(\theta, \phi, V_x, V_y)$$

4D light field

[Adelson and Bergen 91]

Clip slide
V_x
V_y

[Fig. [42]](#)
The 4D light-field is created by eliminating factors from the plenoptic function. Inside the camera, only two measurements needed, in order to create a 4D light-field. The angle of the ray, as it passes through the main lens, is the first measurement (θ,φ,), and the position of the pixel on the camera secondary microlences is the second (Vx, Vy).



Dimension	Viewing space	Name	Year
7	free	plenoptic function	1991
5	free	plenoptic modeling	1995
4	inside a 3D box	Lightfield/Lumigraph	1996
3	inside a 2D circle	concentric mosaics	1999
2	at a fixed point	panorama	1994

Fig. [43]
A taxonomy of plenoptic functions.

Light-field (Plenoptic) Cameras

The popularity of light-field photography has increased in recent years because of the development of hand-held plenoptic cameras. A plenoptic camera is equipped with a micro lens array or printed mask that is placed directly over the camera's imaging sensor. This filters the light that passes through the camera. Based on the properties of the main lens and the filtering device, we can extract light-field rays from measurements on the imaging sensor (Fig [44]).

Each ray is described by its 2D intersection with the main lens and its 2D intersection with the image sensor. The two measurements produce a 4D ray. These rays make up the 4D light field, which can then be processed using integral imaging to create a 2D image. Plenoptic cameras are also referred to as light-field cameras because they capture the lightfield [MDY15].

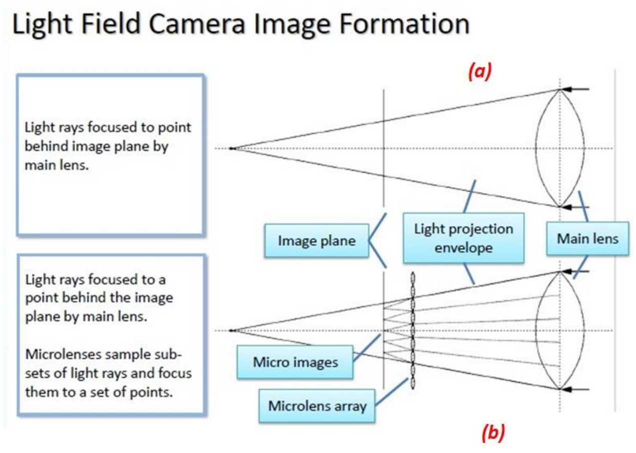

Light Field Camera Image Formation
(a)
Light rays focused to point behind image plane by main lens.
Image plane
Light projection envelope
Main lens
Light rays focused to a point behind the image plane by main lens.
Microlenses sample sub-sets of light rays and focus them to a set of points.
Micro images
Microlens array
(b)

Fig. [44]
Traditional (a) vs. Lightfield camera (b). The majority of cameras that capture 4D light-fields use a micro lens array to modulate the incoming light before it hits the image sensor. Moreover, the sensor uses angle sensitive pixels that measure the angle of the incoming ray, in addition to its energy.

8. Panorama Light-Field Imaging

So far, no light-field capture technique is able to satisfy, both high spatial and high angular requirements. Light-fields that were captured by a camera array used to have high spatial resolution, but very low angular, because of the large baseline between neighboring cameras. In contrast, light-field cameras that can capture at a higher angular resolution (e.g., 14μm in Lytro and 207μm in Raytrix) have low spatial resolution. The problem is inherent to its design of using a 2D sensor to capture a 4D LF: the total number of pixels (rays) that we can capture is fixed and we have to tradeoff between spatial and angular domains [GZCSSC06].

The effective baselines of light-field cameras also tend to be much smaller than those of camera arrays. This is because the effective baseline in an light-field camera is limited to the size of its aperture. Consequently, the captured light-field generally exhibits a much smaller parallax. In practice, when using the light-field camera for post-capture (virtual) refocusing, the user will need to position the camera close to the target object to acquire sufficient parallax for synthesizing noticeable defocus blur effects. As a result, it is difficult to capture the entire object within the camera's FoV and simultaneously generate significant depth-based effects [GYKLY01].

For the aforementioned reasons, light-field photography is still dedicated to capturing single scenes, just like common cameras. Even though it has several advantages over them, in order to appeal to mass market, light-field photography should at least support the imaging techniques that have become standard in conventional photography.

One of these techniques is panorama imaging, which is supported by nearly every modern digital camera or cell phone or can be easily constructed with post processing. However, the application of such well known image processing techniques to plenoptic data is not straightforward, as recorded spatial and directional information is neither independent nor can it be processed in the same way [BIB14]. Nevertheless, there is extensive research that is conducted, and many different approaches have been explored over the years, and we present some of them, as follows:

Fig. [45]
The evolution of light-field cameras. Top the original experimental setups: Stanford Large Statue Scanner (a), Lego Mindstorms Gantry (b). Bottom: Commercial light-field cameras from Lytro and Raytrix.

i. *Naïve Multi-Perspective Image Stitching (2D Image Extraction From 4D Light-Field)*

Naive Multi-Perspective Image Stitching, is the process of stitching corresponding perspective images of the input light fields individually, using classical panorama imaging [BL07]. According to this process 2D images are extracted from raw light-field files, and are post processed in Hugin, just like normal images. We used two sources of publicly available light-field data sets. One from earlier experimental setups and another from a commercial second-generation plenoptic camera (Lytro Illum). We processed these datasets, with Lightfield ToolBox v0.4 [LFT04] or with proprietary software.

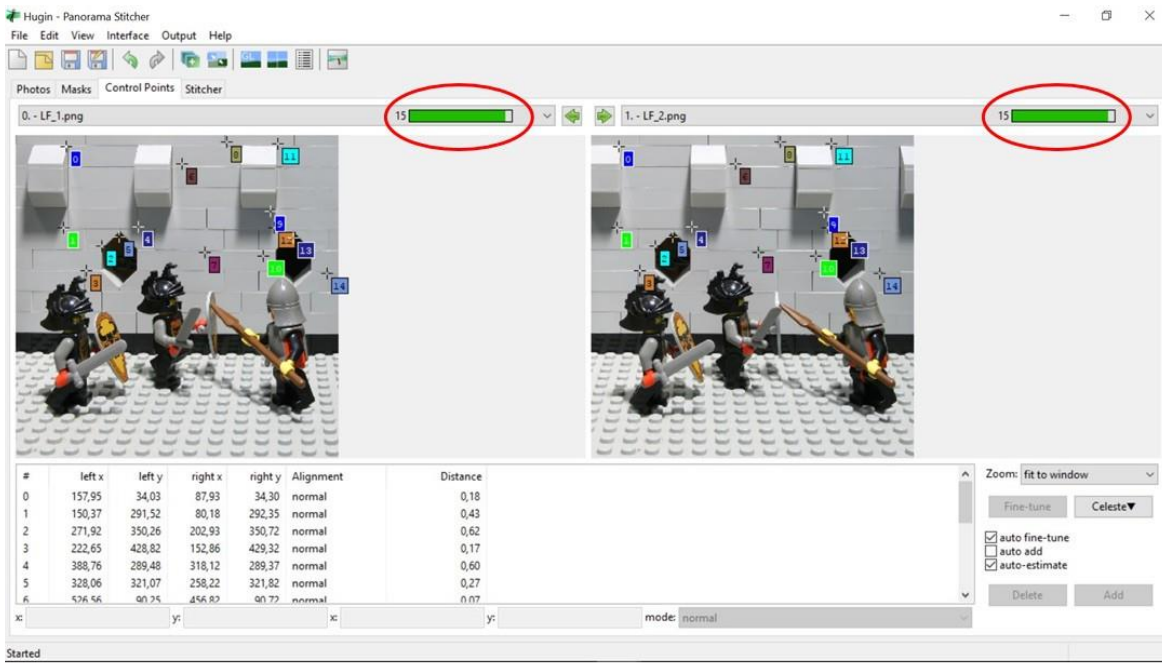
(1) Stanford University's Experimental Setups

Stanford University's Computer Graphics Laboratory was a pioneer in light-field creation. It's first generation light-field setup was custom built by Cyberware Inc. of Monterey in 1999 [LFA01], and was designed expressly for the digitization of large statues, since it concerned the scanning of the sculptures and architecture of Michelangelo [SLSS99]. It was created by mounting commercial of the self dSLRs on a gantry. Needless to say, it was very cumbersome and hard to operate (Fig [45]).

Demonstrating that for capturing light-fields, cumbersome and expensive specialized equipment was not needed, just a sufficient method to move a camera left, right, up and down, a new design [LFA01] was developed by Andrew Adams, who used Lego Mindstorms to build a mounting platform. This mounting platform was equipped with a common dSLR camera, a Canon Digital Rebel XTi, with a Canon 10-22 mm lens. A wide angle lens was used to avoid the need to rotate the camera to keep the scene in view while translating the camera horizontally. The light-field produced, was created by combining 289 normal images, rectified, cropped and arranged in a 17x17 array, and then imported and processed in Light-field ToolBox [LFT04].

Although this two-step process, is vastly different from the process used today, still it is very convenient for choosing suitable images for composites, without any further image processing. We deemed suitable for panorama composite generation, airs of images consisted of the first and the last image of a line or a column in the 17x17 array, and we then imported each pair into Hugin.

Hugin generated a composite with very strong parallax effect and a narrower field of view than a conventional panorama (Fig. [46]). We switched to the advanced interface of Hugin, and it came to our attention, that a 90% overlap (Fig. [47]) between the pair of images, is accounted for the narrow field of view. As for the parallax effect, shifting of the perspective between these two images could not be compensated by the image correcting algorithms. We expanded this technique, with multiple pairs of images, but results were the nearly identical.


Hugin - Panorama Stitcher
File Edit View Interface Output Help
Photos Masks Control Points Stitcher
0. - LF_1.png
15
1. - LF_2.png
15
Zoom: fit to window
Fine-tune
Celeste▼
Delete
Add
Started

#	left x	left y	right x	right y	Alignment	Distance
0	157,95	34,03	87,93	34,30	normal	0,18
1	150,37	291,52	80,18	292,35	normal	0,43
2	271,92	350,26	202,93	350,72	normal	0,62
3	222,65	428,82	152,86	429,32	normal	0,17
4	388,76	289,48	318,12	289,37	normal	0,60
5	328,06	321,07	258,22	321,82	normal	0,27
6	526,56	90,25	456,82	90,72	normal	0,07

[✓ auto fine-tune]
[ ] auto add
[✓ auto-estimate]

mode: normal

Fig. [47]
Even when choosing pairs of images out of the first and the last image of a line or a column, in order to maintain maximum separation, they still had nearly 90% overlap, so a small FoV is expected in the final composite.

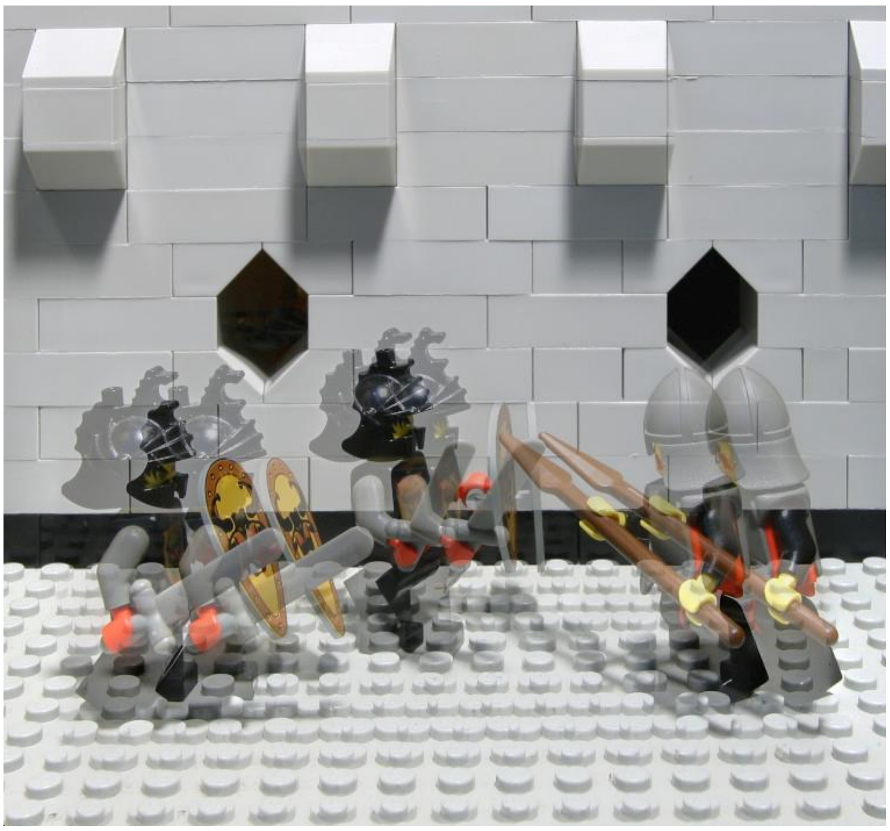


Fig. [46]
Not only the final composite had a narrow field of view, it also presented a very strong parallax effect.

(2) Commercial second generation plenoptic camera (Lytro Illum)

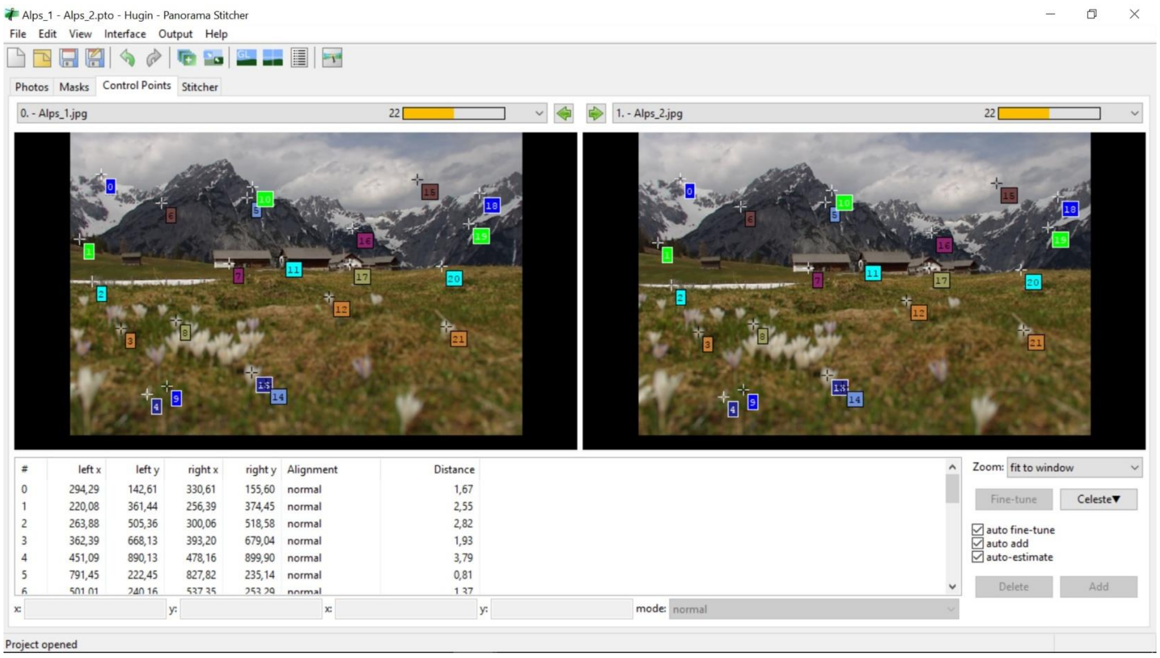
Lytro Illum is a consumer light-field camera [LIR15] with a 30-250 mm lens and a constant f/2.0 aperture, a 40 mega ray sensor and an integrated Snapdragon processor. A proprietary post processing suit for light-field image processing (Lytro Desktop Suite v5.0.1) and various raw light-field files were available for experimentation, without the need to buy the camera itself. An available online viewer was used to extract pairs of images, since the desktop suite is more tuned to depth of field, and color schemes manipulation. Moreover, raw light-field files generated by the camera, were incompatible with Light-field ToolBox. Sadly, Lytro has seized operations since then, and the viewer is not accessible anymore.

In most cases, results were similar to the experimental setups used above, i.e. strong parallax effect and a narrow field of view, pointing out the limits of this technique. It was possible though, to generate a composite, retaining the narrow field of view but without parallax effect. It seems that the perspective sift in the background of the image, was low enough for the image correcting algorithms to compensate (Fig. [48], Fig. [49]).

ii. *Concentric Mosaics*

Concentric Mosaics, are panoramas that contain more than two perspectives. They are captured by slit cameras moving in concentric circles. By recording multiple concentric mosaics at various distances from the rotation axis, novel views within the captured area (i.e., a horizontal 2D disk) can be computed (Fig.[50],Fig.[51]) [SH99].

Concentric mosaics, are a very early implementation of the plenoptic function. Setups are similar in function and construction to Stanford's, but this time camera motion is constrained to planar concentric circles. Concentric mosaics are created by composing slit images taken at different locations along each circle. Concentric mosaics index all input image rays naturally in 3 parameters: radius, rotation angle and vertical elevation. Compared with a light-field, concentric mosaics have much smaller file size because only a 3D plenoptic function is constructed. Unlike panoramas in which the viewpoint is fixed, concentric mosaics allow the user to move freely in a circular region and observe significant parallax and lighting changes without recovering the geometric and photometric scene models.


Alps_1 - Alps_2.pto - Hugin - Panorama Stitcher
File Edit View Interface Output Help
Photos Masks Control Points Stitcher
0. - Alps_1.jpg
1. - Alps_2.jpg
Zoom: fit to window
Fine-tune
Celeste
Delete
Add

#	left x	left y	right x	right y	Alignment	Distance
0	294,29	142,61	330,61	155,60	normal	1,67
1	220,08	361,44	256,39	374,45	normal	2,55
2	263,88	505,36	300,06	518,58	normal	2,82
3	362,39	668,13	393,20	679,04	normal	1,93
4	451,09	890,13	478,16	899,90	normal	3,79
5	791,45	222,45	827,82	235,14	normal	0,81
6	501,01	240,16	537,35	253,29	normal	1,37

☑ auto fine-tune
☑ auto add
☑ auto-estimate

mode: normal

Project opened

Fig. [49]
The same procedure as above but this time, these images are extracted from the online viewer [LIR15].

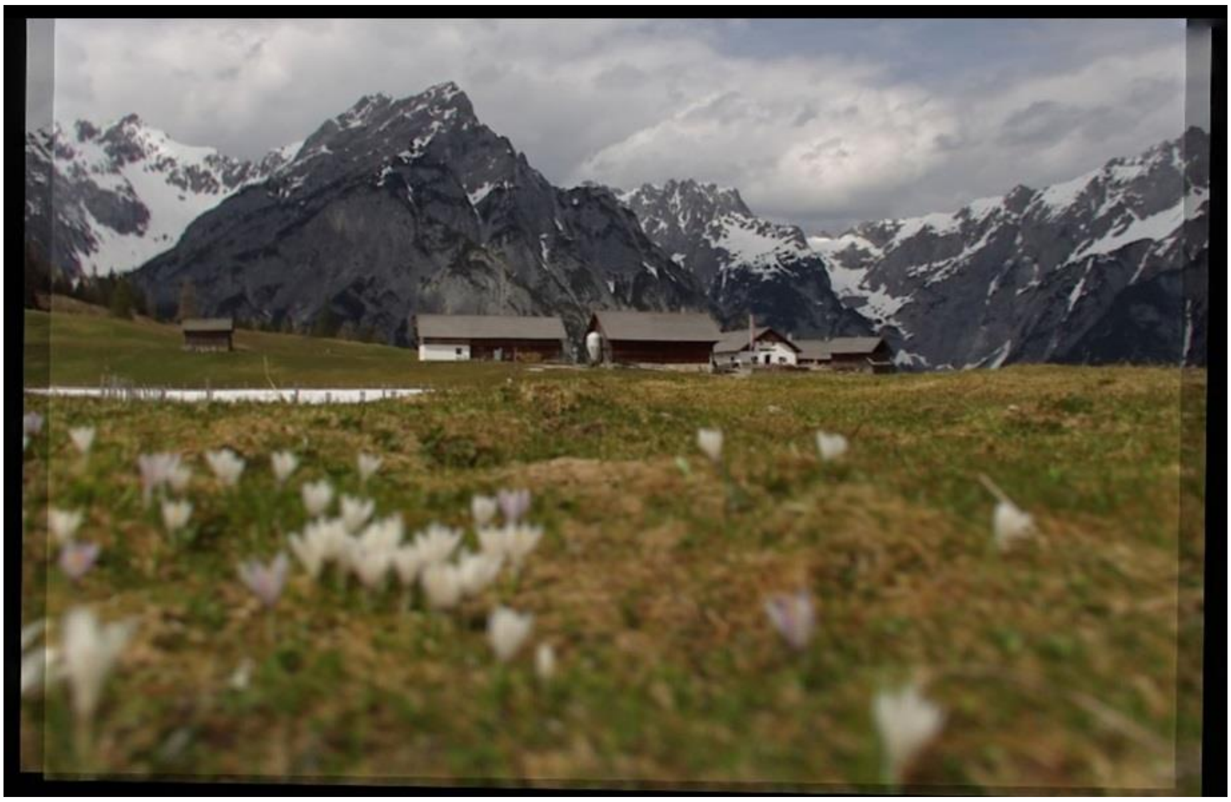


Fig. [48]
FoV is still narrow, but the parallax error is eliminated at least.

Novel views are rendered efficiently by combining appropriate rays that are captured and stored in concentric mosaics. Capturing concentric mosaics is as easy as capturing conventional panoramas except more images are taken, by putting a video camera on a rotary table, and simply spin it slowly around an off-centered circle. A concentric mosaic is also formed by capturing a continuous video sequence.

A critical factor that limits concentric mosaics, is the appearance of vertical distortions while creating them. Small FoV environments should be captured because vertical distortions increase with larger FoVs. Vertical distortions also become more apparent as the user moves backward and forward because significant parallax change occurs. On the other hand, parallax change caused by lateral moves is significantly less, and can therefore be better compensated.

iii. *Light-Field To Focal Stack*

Light Field To Focal Stack, is an approach to constructing and rendering panoramic light fields (i.e., large field-of-view gigaray light fields computed from overlapping, lower-resolution sub-light-field recordings). By converting overlapping sub-light-fields into individual focal stacks from which a panoramic focal stack is computed, we remove the need for a precise reconstruction of scene depth or estimation of camera poses [BOB13].

The main difference from regular image panoramas, is that directional information must be additionally encoded in a consistent way. In contrast to concentric mosaics, these panoramic light fields are captured with regular (mobile) light-field cameras in exactly the same way image panoramas are recorded with conventional cameras (i.e., via rotational movement). A panorama light-field is created by first converting input light fields into focal stacks, stitching these focal stacks, and converting the resulting panorama focal stack into a light-field using linear view synthesis [LD10].

Overlapping sub-light-fields of a scene are captured, by implementing a rotational movement of a mobile light-field camera and converting each sub-light-field into a focal stack using synthetic aperture reconstruction. Next, an all-in-focus image for each focal stack is computed by extracting and composing the highest-frequency image content throughout all focal stack slices.

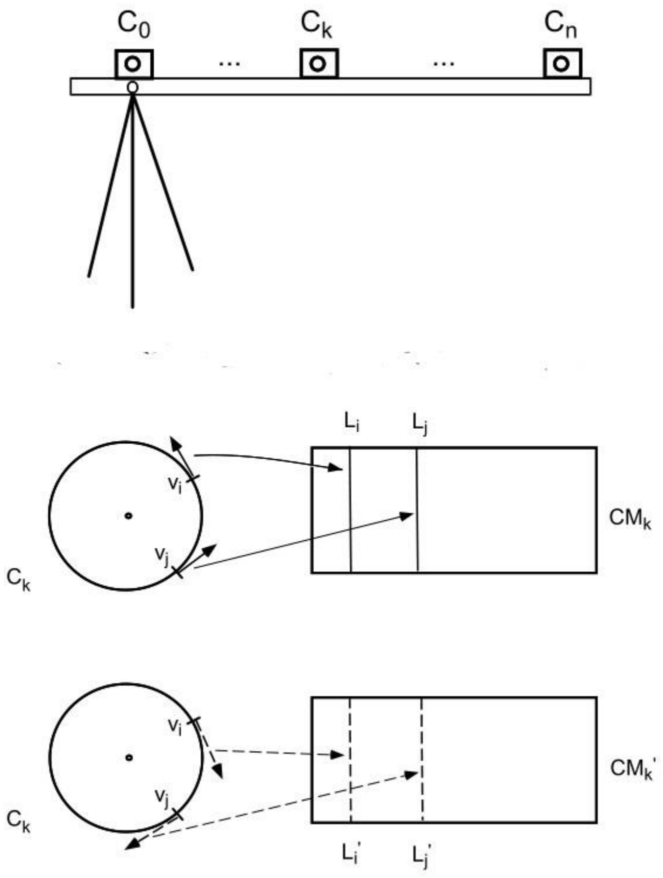


Fig. [50]
An experimental setup and the method, for constructing concentric mosaics.

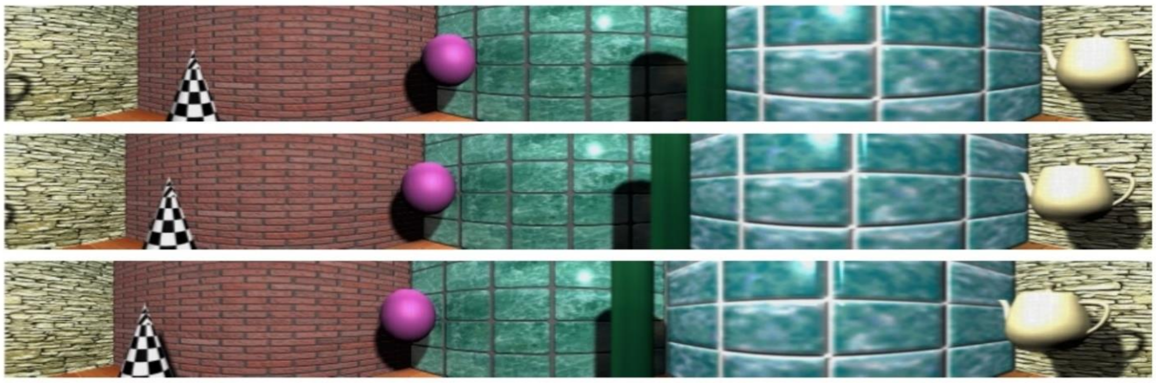


Fig. [51]
A concentric mosaic example.

When conventional panorama stitching techniques are applied to the resulting (overlapping) all-in-focus images, registration and blending parameters are also computed, and then applied to all corresponding slices of the focal stacks. The result is a registered and seamlessly blended panoramic focal stack that can be converted into a light field with linear view synthesis, as described in [LD10]. This process is summarized in figure 52.

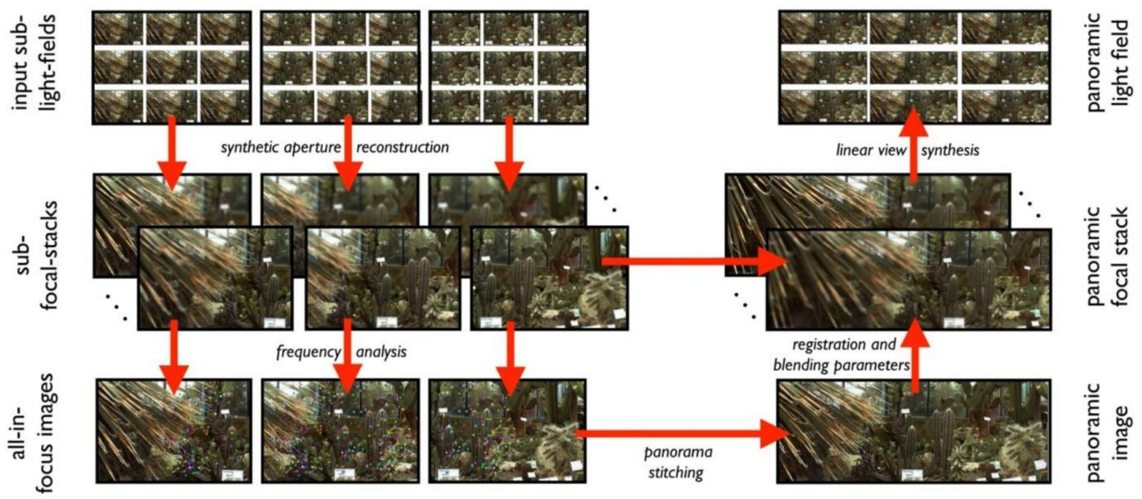


Fig. [52]
The workflow for creating light field composites, from focal stacks. All of the methods of light field panorama imaging that were examined up to now, involve some kind of 4D to 2D transformation, applying known digital panorama imaging techniques, and then reconstructing the light-field. This process is prone to a lot of errors and has many limitations.

All in all is a simple and robust approach, that uses conventional panorama stitching techniques, with no need of precise depth reconstruction or pose estimation and compatible to all generations of plenoptic cameras but also compatible to focal stacks captured by conventional ones. On the other hand, this technique is only applied to surfaces with modest depth discontinuities that have the same radiance when viewed from any angle (Lambertian). Also panorama stitching may distort focal stacks, and cause problems for linear view synthesis (PSF changes).

iv. *Panorama Light-Field Imaging*

In contrast to previous methods, that estimate panorama light fields from focal stacks or from sub light-fields (naive multi-perspective image stitching), this approach is the first that processes ray entries directly and does not require depth reconstruction or matching of image features [BIB14], and it is performed in the following steps:

Plenoptic Camera Parameterization and Calibration. Ray-coordinates in the i, j, k, l form (i.e., micro-image pixel coordinates and lens let indices), are transformed to the form s^0 ,t^0 ,u^0 ,v^0 (i.e., ray intersections on the S^0 T^0 and U^0 V^0 planes), with the help of matrix H^0 [DPW13]. In the end, eight intrinsic parameters that all depend on zoom and focus settings of the camera's main optics, must be determined (w_a ,h_a ,w_f ,h_f ,o_u ,o_v ,d_{uv} , and d_{wx}), so as individual parameter sets must be pre-calibrated for the different zoom and focus settings (Fig. [53]).

Parameterization and Registration of Panorama Light-Fields, where rays are described on nested UV and ST cylinders rather than on parallel planes. (Fig. [54]). Each ray is parameterized by β,h,u,v as opposed to s,t,u,v coordinates, since the horizontal and vertical perspectives are defined by the angle β and the height h and not by an s,t coordinate. Also u, v are coordinates on the UV cylinder. This is a two-step process:

(1) Since parameters of the plenoptic camera for the desired focus and zoom settings of the main optics, are already pre-calibrated, when during camera rotation a set of n input light fields are captured, n+1 registration parameters, must be determined: the distance d_r of the rotated ST planes to a common rotation axis and the individual rotation angles α_i ,i=1..n between each successive input light-field pair. Other parameters, such as the pitch or roll angles of the cameras are assumed to be zero. This is illustrated in (Fig. [55]) for two adjacent input light fields. Inaccuracies of the user and imprecisions of calibration are also corrected by this registration method.

(2) Ray Re-Parameterization and Blending: After all input light fields have been registered, their rays are converted from the individual to every light-field two-plane parameterizations (s,t,u,v) to the common cylindrical parameterization (β,h,u,v) (Fig. [56]).

While the camera is rotated for recording the panorama light field, focus and zoom settings of the main optics of the plenoptic camera are not changed, since the initial calibration. Then the remaining questions are, what are the optimal choices for the distance of the camera to the rotation axis d_r and for the rotation angle α the user should apply and what are the optimal radii of the UV / ST cylinders.

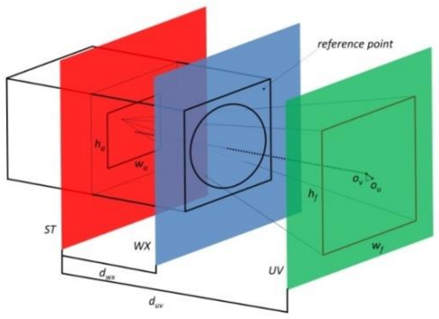


Fig. [53]
Eight intrinsic plenoptic camera parameters: w_a ,h_a ,w_f ,h_f ,o_u ,o_v ,d_{uv} , and d_{wx} . UV and ST are the focal and perspective planes inside and outside the camera housing. WX is the plane that intersects a predefined reference point on the camera housing. All planes are parallel and perpendicular to the camera's optical axis (dotted line). They all depend on zoom and focus settings of the camera's main optics.

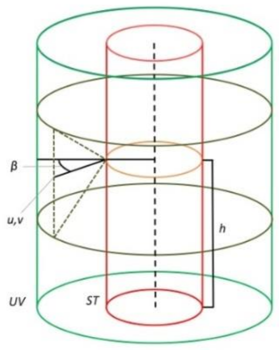


Fig. [54]
Two-plane light-field parameterization (2PP) is converted to cylindrical: Multi view circular / perspective projection over two nested cylinders ST and UV. The horizontal and vertical perspectives are defined by the angle β and the height h. One ray is parameterized by β, h and the intersection with the UV cylinder (u,v).

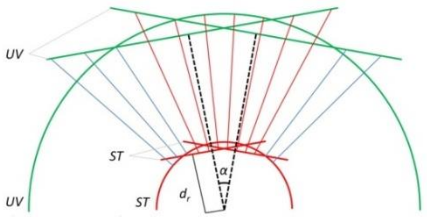


Fig. [55]
Registration of two adjacent input light fields. Choosing a correct set of parameters d_r and α, is integrall in registering two adjastent light-fields, without errors. The overlap between the input light fields should be as large as possible, while the number of recordings should be as small as possible.

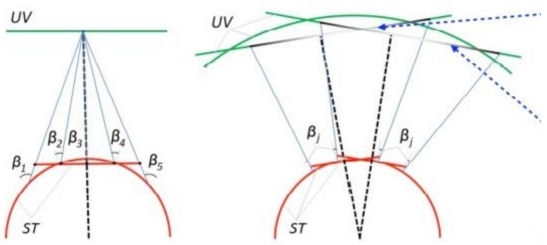


Fig. [56]
Re-parameterization and blending of two adjacent input light fields: Projecting the vertical center line in the V-direction of the UV plane through all horizontal samples in the S-direction on the ST plane of the registered input light field onto the ST cylinder yields all possible horizontal perspective coordinates β_j, while for the vertical perspective h(k) = t(k). All necessary rays of the same input light field are alpha-blended (transparency) with other input light fields for the same β_j.

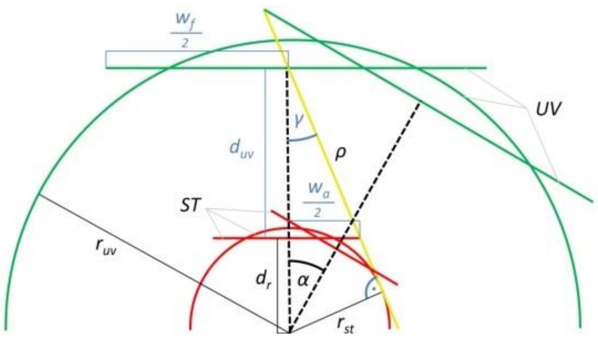


Fig. [57]
Geometric relationship between two-plane parameterization for given intrinsic camera parameters and optimal choice of cylindrical parameterization that fulfills the following constraints: As many of the recorded input rays as possible should be retained without causing empty samples in the resulting panorama light field. The overlap between the input light fields should be as large as possible. The number of recordings should be as small as possible.

Given the relative positions of ST and UV planes of the plenoptic camera (known from intrinsic calibration), these optimal parameters must be determined considering the following constraints: *First*, as many of the recorded input rays as possible should be retained without creating empty samples (spaces) in the resulting panorama light field. *Second*, the overlap between the input light fields should be as large as possible. *Third*, the number of recordings should be as small as possible.

The first is achieved by choosing cylinder radii (r_{uv} and r_{st}) that can be discretized well by the ST and UV planes of the input light fields, the second is achieved by selecting a minimal d_r, and the third can be achieved by maximizing α (Fig. [57], (Fig. [58])).

Plenoptic Camera Parameterization and Calibration. In contrast to methods examined in previous sections, this approach can compute correct panorama light fields that contain all information recorded in the input light fields. However, it also suffers from several limitations:

First, it shares all limitations of classical light-field since only sceneries with near objects lead to useful refocus and parallax effects. For far distant scenes, as usually recorded with classical panorama images, light-field recording would be unnecessary, as of now.

Second, this approach requires a dense ray sampling (small parallax between light-field perspectives) as linear interpolation is applied to compute missing rays during registration and re-parameterization. A larger parallax would lead to blur in the output images.

Third, several assumptions are made in this approach (e.g., the rotation point is on the optical axis, the camera is not rotated around its optical axis, the optical axis is horizontally centered with respect to the camera housing, etc.) that, if strongly violated, can lead to failures, since they affect initial calibration. Also fabrication imprecisions of the panoramic tripod head can, for instance, cause slightly different camera poses when stepping through panorama capturing. This leads to blur when blending multiple light-field recordings.

$w_a, h_a, w_f, h_f, o_u, o_v, d_{uv},, d_{wx}, d_r$ in [mm], α in [°]										
zoom	w_a	h_a	w_f	h_f	o_u	o_v	d_{uv}	d_{wx}	d_r	α
1.0×	2.66	2.66	182.40	182.40	2.59	3.38	263.07	20.00	3.00	16.28
1.5×	4.43	4.43	254.23	254.23	0.25	9.14	529.40	31.00	3.67	11.33
2.0×	5.71	5.71	336.85	336.85	0.41	2.14	976.24	40.00	4.15	7.70

Fig. [58]
Calibrated intrinsic and optimal recording parameters of Lytro plenoptic camera in everyday mode (default refocus range) for three different zoom steps with $\alpha = 2^o$ and $d_r = 2$mm.

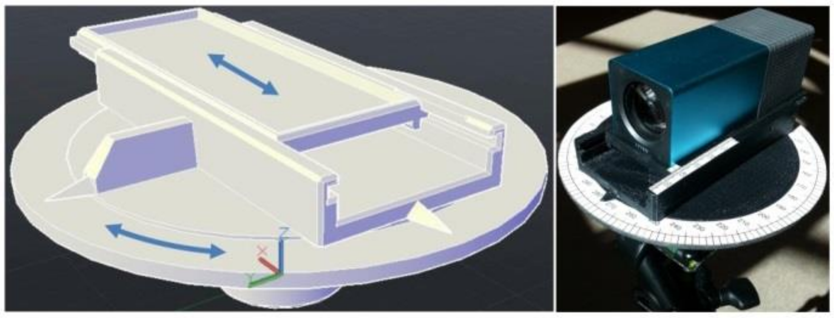


Fig. [59]
Panoramic tripod head: construction drawing (left) and 3D print (right). The adjustable slide allows manual α-rotations and d_r-shifts

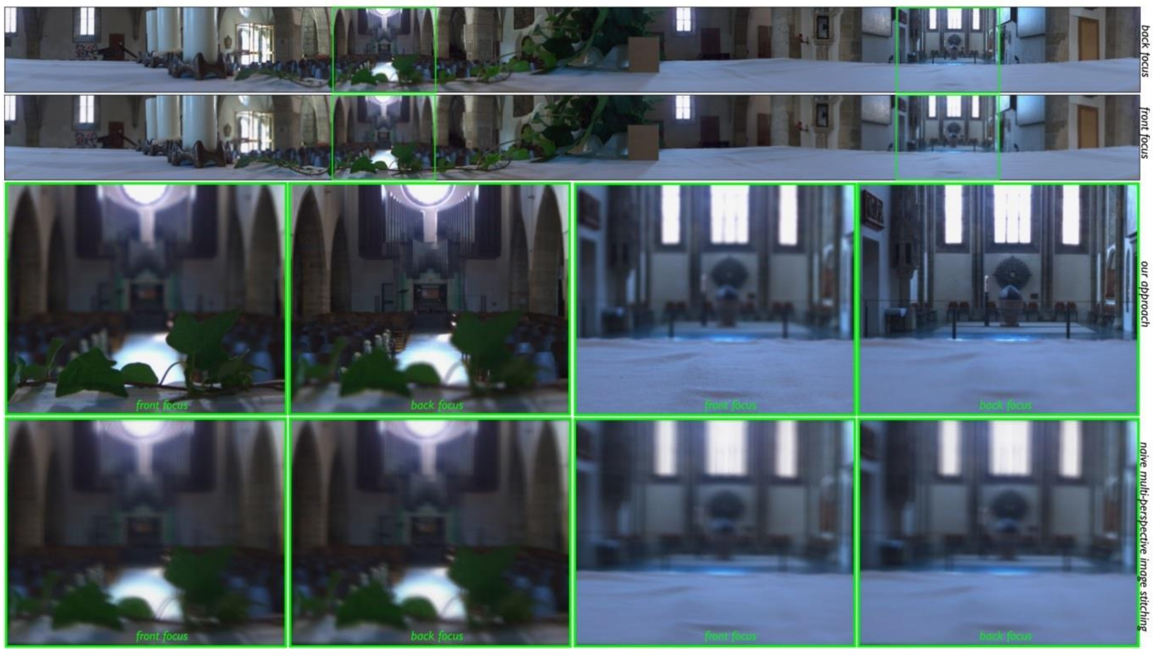


Fig. [60]
Comparison of this method with a naive multi-perspective image stitching of input light fields: Since the spatial and directional domains are processed independently, the resulting panorama light field is inconsistent in the directional domain. This leads to strong artifacts in rendered images, in particular for re-focusing (bottom).

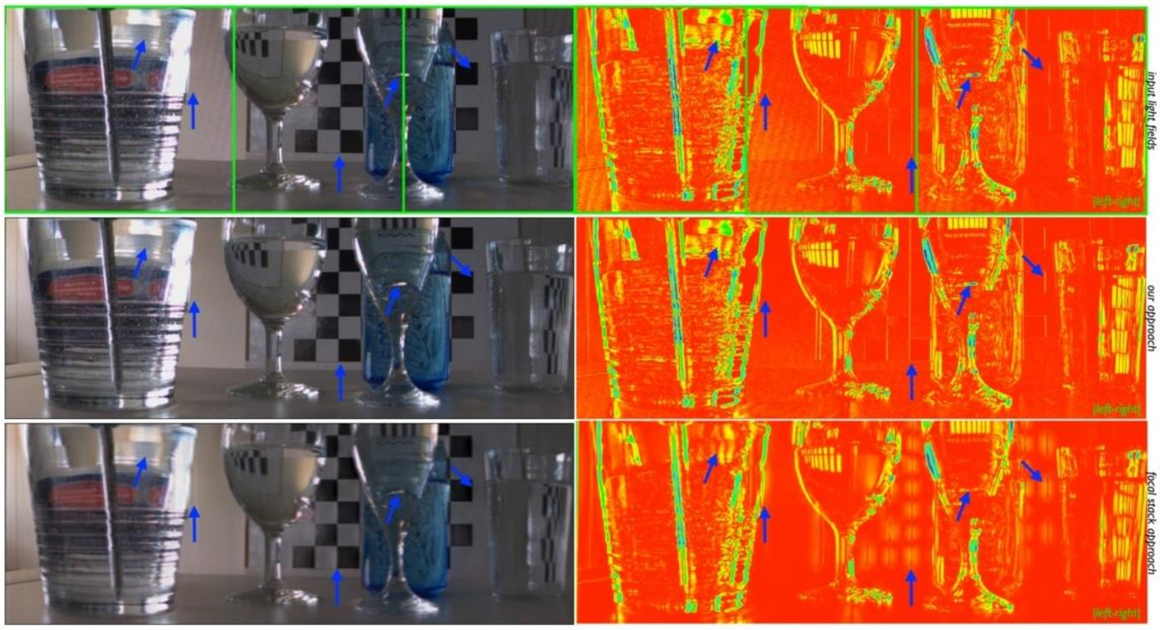


Fig. [61]
Comparison the focal stack approach: The overlaid perspectives of the original input light fields (top) display the recorded scenery and light effects that should be retained in the resulting panorama light field (center and bottom). This is the case for this method (center). The focal stack approach (bottom), however, does not preserve correct anisotropic reflections and refractions and causes artifacts at occlusion boundaries. The blue arrows indicate some of these errors. The color coded difference images (right) illustrate the variation between left-most and right-most light field perspectives.

v. *Enhancing Light Fields Through Ray-Space Stitching*

This is a a light-field enhancing technique that merges multiple light-fields, that are captured with a common light-field camera. This is analogous to stitching multiple 2D images into a panorama. It depends on calculating new matrix called ray-space motion matrix (RSMM) that describes how light-field ray parameterization are transformed under different light-field coordinates.

This approach is closest to that of section 8.iv (Birklbauer and Bimber [BIB14]), since it also acquires and fuses multiple light-field, but as mentioned above it targets at registering and transforming multiple rotated two-plane parameterized (2PP) light fields into a global cylindrical coordinate system (composed of two nested cylinders rather than parallel planes), that requires the camera moving precisely during light-field capturing in order to minimize registration artifacts.

In contrast, this approach allows the user to freely rotate or translate the light-field camera and aligns two 2PP light-field into a common 2PP parameterization through matrix transformation. A high dimensional graph cut is then conducted to compensate for misalignments.

Overview of this technique: Light-fields are captured in sequence, and the first one is used as the reference. A 5×6 matrix is sufficient to transform rays from one light-field to another. This matrix is defined as the ray-space motion matrix (RSMM). The RSMM is computed for all adjacent light-fields and by chaining RSMMs all light-fields can then be transformed to the reference coordinate system.

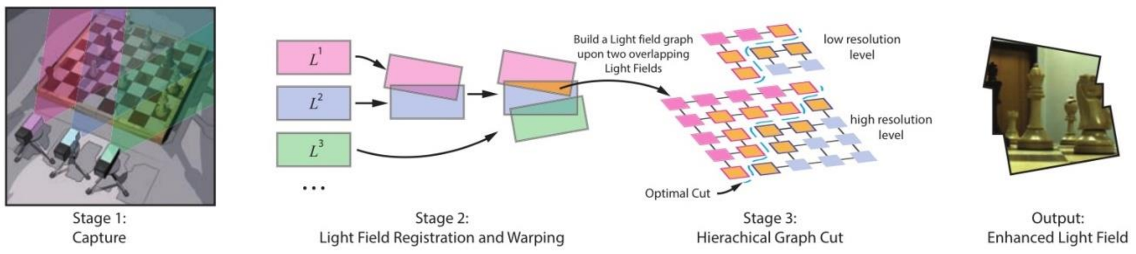


Fig. [62]
The workflow of this light-field stitching algorithm. Each 4D light-field is presented as a 2D plane for ease of visualization.

At each iteration after the two light-fields are aligned, the stitching process is refined since it is necessary to account for slight errors in estimating the RSMMs and to handle slight scene motion, since the light-fields were sequentially captured. To stitch each pair of light-fields, their overlapping ray subspace are mapped to a 4D graph and rely on hierarchical graph-cut to efficiently find a seamless boundary. This process is summarized in Fig. 62.

Ray-space motion matrix: The conventional two-plane parameterization (2PP) is used to represent a light-field. Each ray r is parameterized by its intersection points with two planes: u,v as one intersection with the sensor plane Π_{uv} , and s,t as the other intersection with the (virtual) camera plane Π_{st} . The two planes are a fixed distance apart, since all light-fields were acquired using the same light-field camera with a fixed configuration. More over the transformation between light-fields is simply determined by the change (rotation and translation) of the 2 Parameterization Planes.

Light-fields alignment, is started with the first pair of light-fields (first and second) and iteratively go through all of them, in order to estimate a pair wise warping function (rotation and translation) and subsequently align them with regard to the first light-field. The 5 × 6 matrix transform that determines the warping between the light-field (Fig.

3.), is defined as Ray Space Motion Matrix (RSMM) and has 21 non-zero entries.

To compute the RSMM between the pair of light-fields a minimum of 6 pairs of ray correspondences is needed, but a much larger set of makes the technique more robust. Large rotations between two light-fields must be avoided, since in a large rotation a ray in the original parameterization may be parallel to the new parameterization plane. (The transformation also has a singularity where γ is zero). On average, it took slightly under 1.5 minutes to extract the RSMMs for two light-fields with a resolution of 10×10×800×800 on a 3.2 GHz CPU.

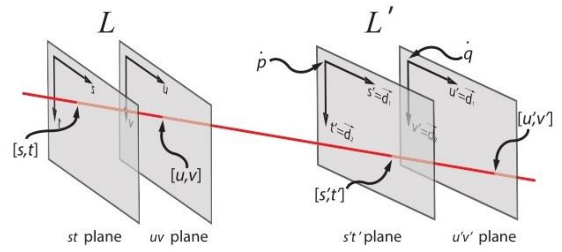


Fig. [63]
Each ray [s,t,u,v] from L, is mapped in the reference L′ [s′,t′,u′,v′], with the use of RMSS. By applying the RSMM transformation, two light-fields are registered under a same 2PP coordinate.

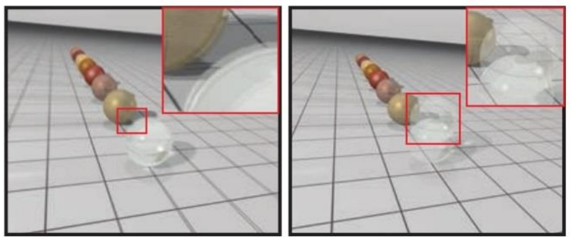


Fig. [64]
RSMM transformation errors: Even though these light-fields have large overlapping subspace, the rays do not match exactly due to under sampling (left). The scene may have changed a little over time and ghosting artifacts due to motion were introduced (right).

Seamless stitching: After applying the RSMM transformation, a pair of light-fields are registered under a same 2PP coordinate. The stitching of two light-fields is then performed by, mapping their overlapping ray subspace to a 4D graph, and then searching for an optimal 4D cut in the overlapped region, that minimizes the inconsistency along the cutting boundary. This problem can be solved precisely by graph-cut. The key observation here is that to reliably stitch two light-fields, we need to measure the differences of adjacent rays in both spatial and angular dimensions.

For efficiency reasons, a coarse-to-fine version of graph-cut [PVT05] is opted. First a graph at a low resolution is constructed and the cut is computed. Then an interpolated cut is used on one level finer, unnecessary nodes are eliminated in the higher resolution graph and finally a new cut is also computed.

Comparisons to other approaches: In the method previously presented, (S8.iv - Birklbauer et al. [BIB14]) the light-field camera is required to rotate precisely around an axis parallel to the sensor. The

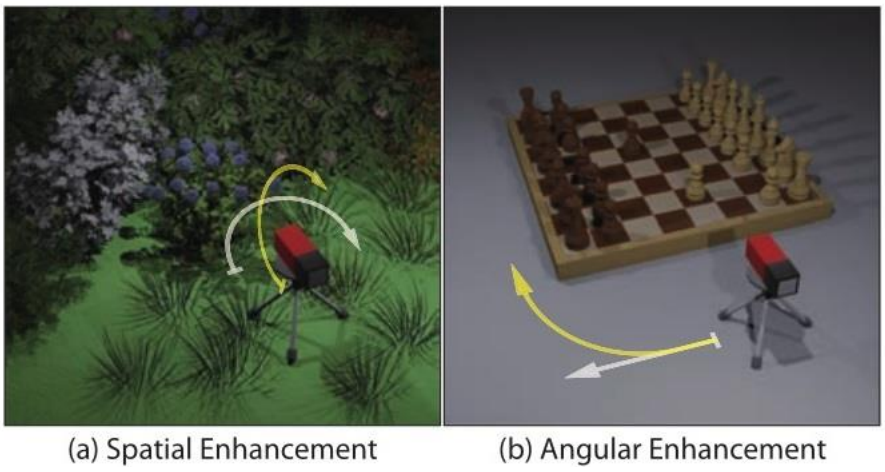


Fig. [65]
How the camera acquires light-fields: (a) The light-field camera is panned and tilted in order to increase the FoV. (b) To enhance synthetic aperture and parallax, light-field camera is translated (white arrow); To enhance rotational parallax, the camera is rotated around the object.

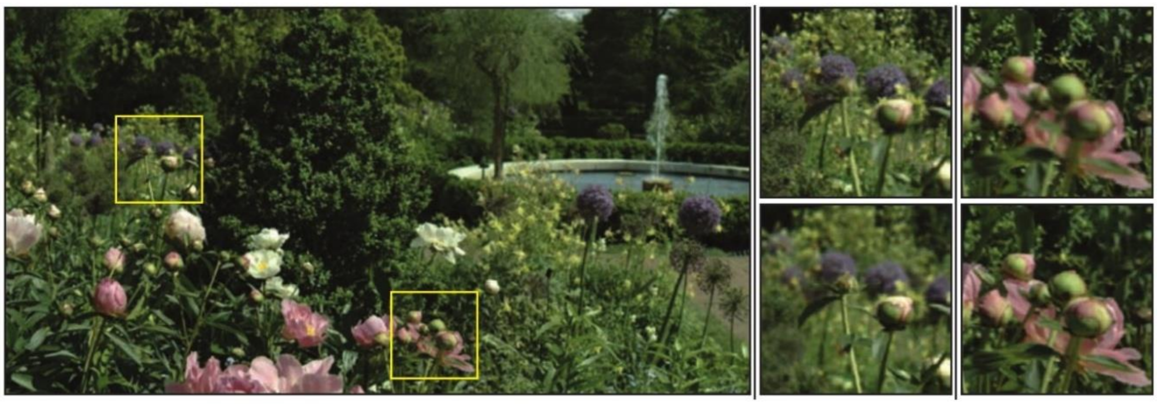


Fig. [66]
A panoramic light-field generated from a 5×4 grid of light-fields. Left: full view focusing at a plane close to the foreground flower. Right 2×2 images: close-up views of highlighted regions focused at the background (top row) and foreground (bottom row).

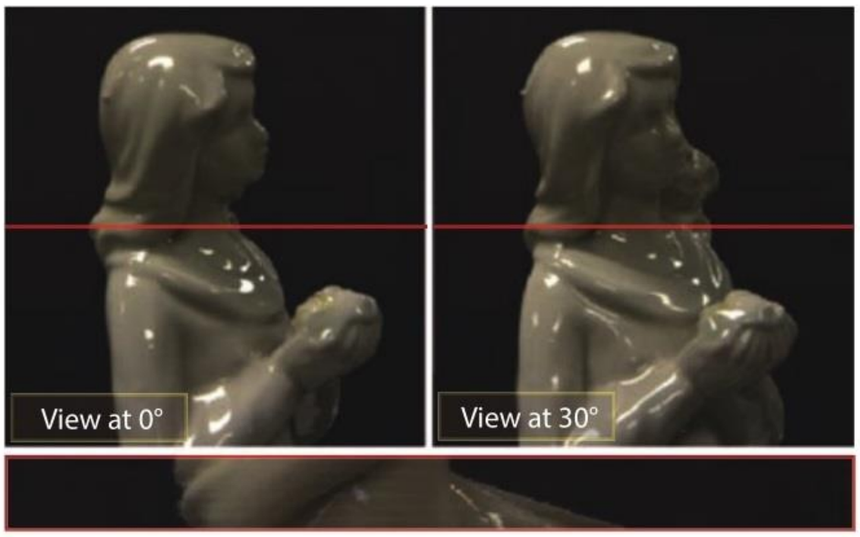


Fig. [67]
A stitched light-field with increased rotational parallax. The top row shows two views from the stitched light-field. The horizontal strip at the bottom is a composite of the profile along the red line as the object is virtually rotated. The strip is the coherent, with no errors.

rotation axis should also intersect with the camera's optical axis and the sequence should be acquired with an approximately identical rotation angle.

In contrast, this technique supports much more flexible camera movement, since capture sequence can exhibit combinations of rotational and translational motions. This is because the general light-field registration matrix RSMM can simultaneously handle rotation and translation. For comparison, the source code [BIB14] provided by the authors was used to stitch light-field data, that were captured by manually rotating the light-field camera where the rotation axis exhibits slight translations across them. Such translational motions, even though very small, violate the assumptions in [BIB14].

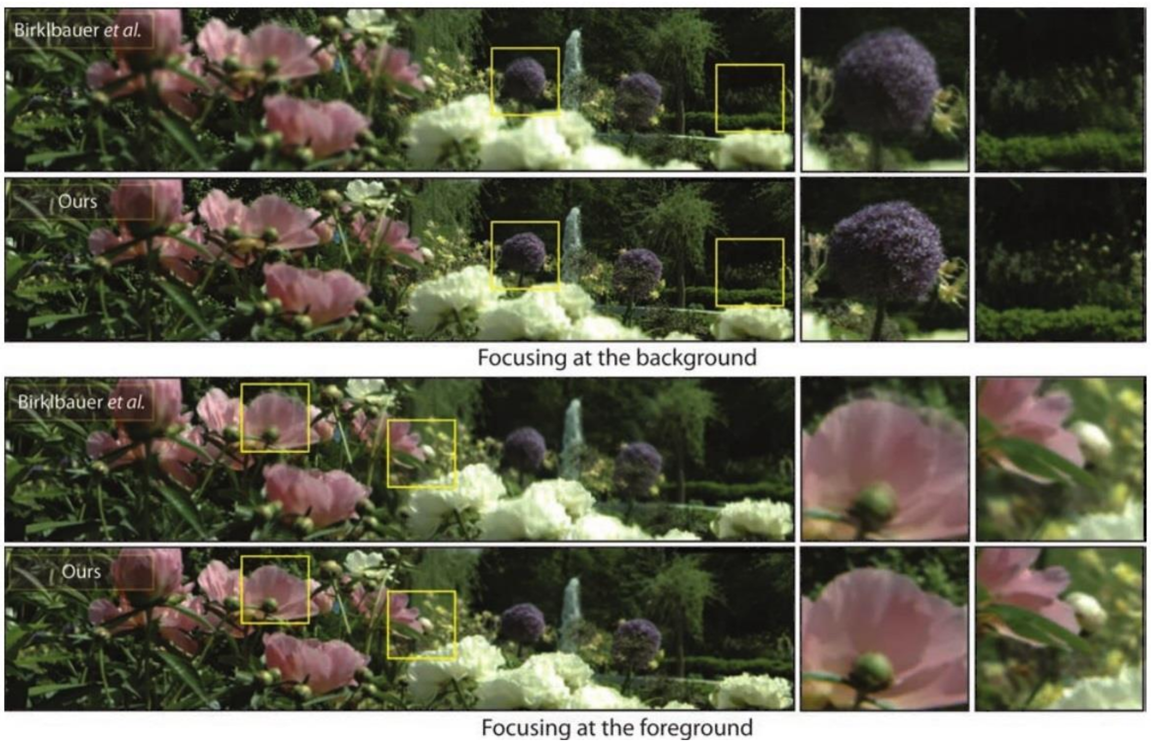


Fig. [68]
Comparisons of light-field rendering using RMSS method vs. the method described in S8.iv, using the same light-field data. RMSS method is able to stitch the input into a ghosting free panoramic light field while the method described in S8.iv produces strong ghosting artifacts due to translational motions across the input light-fields.

As a result, the refocused results exhibit strong ghosting (aliasing) artifacts due to misalignment. On the contrary with this approach, results are nearly aliasing free, preserve sharp details, and more importantly, frees the user from precise light-field acquisition. Fig. 68 compares the refocused rendering results on the stitched light-fields produced by these two methods.

Failure cases: Adjacent light-field need to have large overlaps, to ensure reliable RSMM estimations. If the estimated RSMM contains small errors, the graph-cut can still effectively eliminate inconsistency and produce visually pleasing results. In this case, however, the stitched result is not a "real" light-field: corresponding rays are not guaranteed to intersect at common 3D points. Fig. 69

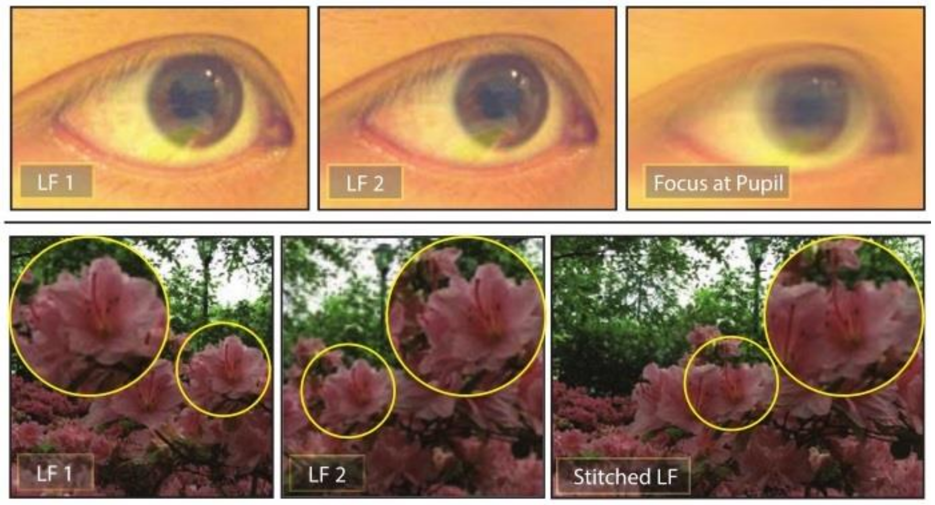


Fig. [69]
Failure cases. Top row: the depth-of-field rendering on the in-focus region (the pupil) exhibits blurs due to errors in RSMM estimation. Bottom row: the flower is incorrectly stitched due to large displacement between the two light-fields.

9. Concluding Remarks

The process of constructing digital panoramic composites, when 2D images are used, has come a long way from its first tentative steps. Instead of searching only for corners, nowadays algorithms have been created that can combine with ease, whole series of overlapping images, into one wide field of view composite. Combined with the rising with ease power of today's pcs and smartphones, this process is only limited, by the experience of the user (and sometimes not even that). It's a mature, readily available and ever evolving technology.

Techniques for combining light-fields into a wide field composite, on the other hand, have a lot of challenges to overcome. Most of these techniques relied on algorithms first applied to 2D composites, so a lot of errors and artifacts were generated in light-fields construction and deconstruction. But by approaching light-fields in a deeper level, akin to using assembly language in order to program a computer, and focusing on coordinates transformation, different light-fields can be combined into a new lightfield, that has a wider field of view but also retains its original properties. These techniques are so promising that in a few years, we may see them applied to commercial products (i.e. full face and body recognition and identification products).

Even so, light-fields still have an inherent constrain: They still cannot capture far field objects, in services like , the way 2D imaging can. Research is being conducted [MDY15], that could allow light-field cameras to be mounted on airborne platforms, and capture far field views, but there is still work to be done, if light-field imaging wants to become as efficient as traditional 2D imaging.

Section A - References

[PP12]

"Panoramic Photography"
https://site.douban.com/195886/widget/notes/11812927/note/252569163/

[BPG07]

IATH Best Practices Guide to Digital Panoramic Photography

http://www2.iath.virginia.edu/panorama/section4.html#4.1

[FDD14]

OpenCV Feature Detection and Description

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

[MRV80]

Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Moravec H, Tech Report CMU-RI-TR-3, Carnegie-Mellon University, Robotics Institute, September 1980.

[CH88]

A Combined Corner And Edge Detector, Chris Harris & Mike Stephens, Plessey Research Roke Manor, United Kingdom, The Plessey Company pic. 1988

[JS94]

Good Features To Track, Jiambo Shi, Computer Science Department, Cornell University, Carlo Tomasi, Computer Science Department, Stanford University , IEEE Conference on Computer Vision And Pattern Recognition (CVPR94), Seattle June 1994.

[DGL04]

Distinctive Image Features From Scale-Invariant Keypoints, David G. Lowe, Computer Science Department, University of British Columbia, Vancouver, B.C., Canada, lowe@cs.ubc.ca, January 5, 2004

[TL93]

Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A Method for Focus-of-Attention, Tony Lindeberg, Computational Vision and Active Perception Laboratory (CVAP) Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden, International Journal of Computer Vision, vol.11, no.3, pp. 283-318, 1993.

[SSI07]

Scale Space of an Image

https://en.wikipedia.org/wiki/Scale_space

[SKF08]

https://en.wikipedia.org/wiki/Scale-invariant_feature_transform#/media/File:Sift_keypoints_filtering.jpg

[IG10]

https://en.wikipedia.org/wiki/Image_gradient#/media/File:Gradient2.svg

https://en.wikipedia.org/wiki/Image_gradient#/media/File:Intensity_image_with_gradient_images.png

[BTG09]

SURF: Speeded Up Robust Features, Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, ETH Zurich {Bay, Vangool}@vision.ee.ethz.ch, Katholieke Universiteit Leuven {Tinne.Tuytelaars, Luc.Vangool}@esat.kuleuven.be (2009)

[BSM20]

Lecture 13: Edge Detection, Bryan S. Morse, Brigham Young University, 1998–2000

[18]

SURF: Speeded Up Robust Features, CRV Tutorial Day 2010,David Chi Chung Tam, Ryerson University

[SR05]

Image Alignment and Stitching, Szeliski, Richard (2005)

[SLW07]

Photographic Stitching With Optimized Object And Color Matching Based On Image Derivatives, Simon T.Y. Suen, Edmund Y. Lam, and Kenneth K.Y. Wong, Department of Electrical and Electronic Engineering,The University of Hong Kong, Pokfulam Road, Hong Kong (2007)

[PT13]

http://panotools.sourceforge.net/

[HD19]

http://hugin.sourceforge.net/

[DS16]

https://www.dronesolutions.gr/blog.php

[VL19]

https://www.videolan.org/vlc/index.el.html

[PPS13]

A Comparison of SIFT and SURF, P.M. Panchal, S.R. Panchal, S.K. Shah, International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 2, April 2013.

[VVS12]

Automatic Image Registration using SIFT-NCC, ViniVidyadharan, and SubuSurendran, Special Issue of International Journal of Computer Applications (0975 – 8887) , pp.29-32, June 2012.

[RS06]

Image Alignment and Stitching - A Tutorial, Richard Szeliski, Microsoft Research, 2006

[MF1846]

Thoughts on Ray Vibrations, Michael Faraday, Experimental Researches in Electricity", Vol III, M. Faraday, p447-452, also Philosophical Magazine, S.3, Vol XXVIII, N188, May 1846

https://www-spof.gsfc.nasa.gov/Education/wfarad1846.html

[AG1939]

The Light-Field, Andrey Gershun, State Optical Institute, Leningrad, 1939 Translated by Parry Moon & Gregory Timochenco, Massachusetts Institute of Technology, 1939

https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm193918151

[MDY15]

Development of airborne light field photography, Michael Dominick Yocius, University of Iowa, Spring 2015

[RMD16]

Principles of Light Field Imaging, Briefly Revisiting 25 Years of Research, Ivo Ihrke, John Restrepo, and Loïs Mignard-Debise, IEEE SIgnal ProcESSIng MagazInE, September 2016, p.59 - p.69

[WHON97]

Image Based Rendering With Controllable Illumination, T. Wong, P. Heng, S. Or, and W. Ng., In Proceedings of the 8th Eurographics Workshop on Rendering ,pages13–22, St.Etienne, France,June1997

[MB95]

Plenoptic modeling: An Image Based Rendering System, L. McMillan and G. Bishop, ComputerGraphics(SIGGRAPH'95), pages39–46, August 1995.

[Che95]

QuickTime VR – An Image Based Approach to Virtual Environment Navigation, S.E. Chen, Computer Graphics (SIGGRAPH'95), pages 29–38, August 1995.

[SS97]

Creating full view panoramic image mosaics and texture-mapped models, R. Szeliski and H.Y. Shum, Computer Graphics (SIG-GRAPH'97), pages 251–258, August 1997

[GZCSSC06]

Spatio-Angular Resolution Tradeoff in Integral Photography, Todor Georgeiv, Ke Colin Zheng, Brian Curless, David Salesin, Shree Nayar, and Chintan Intwala (2006), Adobe Systems, University of Washington, Columbia University

[SH99]

Rendering Concentric Mosaics, Heung-Yeung Shum, Li-Wei He (1999), Microsoft Research

[BOB13]

Rendering Gigaray Light Fields, C. Birklbauer, S. Opelt and O. Bimber (2013), Institute of Computer Graphics, Johannes Kepler University Linz, Austria

[LD10]

Linear view synthesis using a dimensionality gap light field prior, A. Levin, F. Durand, In Proc. IEEE Computer Vision and Pattern Recognition (2010), pp. 1831 –1838. 5

[BIB14]

Panorama Light-Field Imaging, C. Birklbauer and O. Bimber (2014), Institute of Computer Graphics, Johannes Kepler University Linz, Austria

[GYKLY01]

Enhancing Light Fields through Ray-Space, Xinqing Guo, Zhan Yu, Sing Bing Kang, Haiting Lin, and Jingyi Yu.

[LFT04]

Light Field Toolbox v0.4

http://www.mathworks.com/matlabcentral/fileexchange/49683-light-field-toolbox-v0-4

[SLSS99]

The Stanford Large Statue Scanner, http://graphics.stanford.edu/projects/mich/mgantry-in-lab/mgantry-in-lab.html

[LFA01]

The (New) Stanford Light Field Archive, Computer Graphics Laboratory, Stanford University, http://lightfield.stanford.edu/acq.html

[LIR15]

Lytro Illum: Review after Three Weeks of Testing

http://lightfield-forum.com/2015/06/lytro-illum-review-after-three-weeks-of-testing/

[PVT05]

Panoramic video textures, A. Agarwala, K.C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, SIGGRAPH, 2005.

[DMB97]

Recovering High Dynamic Range Radiance Maps from Photographs, Paul E. Debevec, Jitendra Malik, University of California at Berkeley, 1997

[DPW13]

Decoding, calibration and rectification for lenselet-based plenoptic cameras, Dansereau D. G., Pizarro O., Williams S.B., In Proc. IEEE Computer Vision and Pattern Recognition (2013).

Section B - Figures

Cover

http://www.somedayillbethere.com/2012/05/three-things-in-the-sand-giza-pyramids/

https://www.instagram.com/cosmic_background

Fig. [1]

A 19th century panorama. “Wharf at Yonkers”, watercolor, Samuel Colman’s 1869-1870.

(https://www.nytimes.com/2013/02/17/nyregion/a-review-of-the-panoramic-river-at-the-hudson-river-museum.html)

Fig. [2]

A panorama of Melbourne's Yarra River at twilight,

Taken on 26 August 2005, by David Iliff, from Melbourne, Australia. For stitching the image, David used PTGui. Less daylight, and many reflections on the water, make this picture a very good example of digital stitching. Fig. 1&2 display a common thematic thread, even though the crafting methods are vastly different.

Fig. [3]

Evolution of an algorithm: Moravec described his Corner Detector Algorithm, and Harris - Stephenson transformed it into an equation. It basically finds the difference in intensity for a displacement of (u,v) in all directions.

Fig. [4]

The different approaches of Harris - Stephenson and Shi - Tomasi. The utilization of λ1 and λ2, λ1 and λ2 being the eigen values of the second-moment matrix M, is the main difference between these two algorithms.

Fig. [5]

Corner Detection using Matlab and the Harris - Stephenson algorithm. corner(), is a built in function of Matlab , that can perform corner detection using both variations of the algorithm, as parameters.

Fig. [6]

Corner Detection using Matlab and the Shi - Tomasi algorithm corner detection, using the same script as Fig[5]. Even though the algorithm perform equally well in both cases, Shi - Tomasi is set as the default variation in the Matlab function. The difference in speed in a common desktop PC is negligible.

Fig. [10]

The idea behind Scale Space generation, by applying Difference of Gaussian (DoG): the subtraction of one blurred version of an original image from another, less blurred version of the original. After applying a Gaussian filter of different variance to the original image, the blurred versions are created.

Fig. [9]

The actual Scale Space generation, as applied to a test image. Different scales t, are generated after filtering the image, with filters of different variance. In the bottom right sub image, regions with the same characteristics are prominent.

Fig. [11]

Keypoint localization and filtering. Left : Location of possible keypoint . Center: Filtering low contrast and those located on the edges candidates. Right: Remaining keypoints that will be processed, in the next step.

Fig. [12]

Keypoint descriptors orientation and generation. Gradients: Blue arrows indicate the direction of the gradient. Dark areas represent higher values.

Fig. [13]

Keypoint descriptors orientation and generation. After computing image gradients , in a region around the keypoint location, a Keypoint descriptor is generated. Right:A 2x2 descriptor array computed from an 8x8 set of samples.

Fig. [14]

SURF reduces computation time significantly, since it calculates the sum of pixel intensities in a rectangular region. Only 3 additions needed, and calculation time is independent of the size [18].

Fig. [15]

SURF works like a blob detector. In the image above, it detects the white blobs on wings of the butterfly.

Fig. [16]

SURF detection, using the OpenSurf implementation in MatLab. Circles represent interest points that are detected within the image. The size of them represent scales. Green lines represent orientation, and the color of the circles (red or blue), denotes bright blobs on dark backgrounds (Red), or dark blobs on bright backgrounds (Blue).

Fig. [17]

SHIFT vs SURF comparison

(a) The original image used as a "test bench". (f) Detected features using SURF. (c) Detected features using SHIFT. It's a tradeoff between speed and the number of features detected. SHIFT can reveal many more features than SURF does, but SURF can perform the same task as SHIFT at nearly half the time.

Fig. [18]

Comparison of various cylindrical panoramic formats. VFoV: 165 degrees HFoV: 360 degrees. There is no distortion in the vertical, but long straight lines are bended in the horizontal plane.

Fig. [19]

Blending images with different methods. (a) uses a simple averaging and (b) applies a median filter. (c) Weighted averaging weight pixels near the center of the image more heavily and to down-weight pixels near the edges (feathering). (d) One way to improve feathering is to raise the values to some large power, and the weighted averages become dominated by larger values (p-norm weighting). (e) Assigning each pixel to the nearest image center (Voronoi diagram).

Fig. [20]

Registration, alligment and blending of multiple images. Red color denotes the seams between the images.

Fig. [21]

Hugin Simple Interface. This part of the interface is designed to be easy to use and straightforward. Castle of Chora series of images, are loaded in order to be processed. (a) Panorama preview window. Left and right images are rotated in order to be aligned with the central image. (b) Lens type and characteristics. These values are usually extracted from EXIF metadata. (c) Generated control points. They are created by the use of feature detection matching algorithms, in the three images.

Fig. [22]

Hugin is switched to its advance type of interface. Left and center images, that will be joined to compose the digital panorama, are loaded. The numbered colored rectangles are control points that were generated after processing common features between these images. Even if this procedure is automatic, there are tools to further refine their positioning.

Fig. [23]

Looking for common features between the above images, using a MatLab SURF implementation (SURF_Detection_Matching.m). Some of the detected common features, after several processing iterations will evolve to control points generated used by Hugin.

Fig. [24]

A new digital composite (panorama). The castle itself, is a series of mansions forming a defense structure.

Fig. [25]

This time eight overlapping images were shot in sequence, using a dSLR and a mounting tripod. They were loaded in Hugin (simple interface pictured) in order to create a panorama of the Livadi Bay. Since the camera was mounted to the tripod, no distortions or parallax errors were noticed.

Fig. [26]

Livadi Bay is located on the eastern part of Astypalaia Island. This is the west shore of the bay.

Fig. [27]

Kilindra Inlet, lies on the eastern shore of Livadi Bay. It was late morning at the time of capture, with the sea at the foreground. Different shades of blue, represent different reflecting qualities of the sea water.

Fig. [28]

The part of the image that is blurred, due to a parallax error. It seems that the camera was unconsciously moved, at the time of capture, pointing at the significance of using mounting equipment, when trying to create complex scenes.

Fig. [29]

This composite was created with an iPhone v5, in a single take without combining different images. Image post processing, was also done on the phone.

Fig. [30]

It seems that straight line image objects such as this road, is appearing curved. This points to incorrect image projection (cylindrical instead of planar/rectilinear).

Fig. [31]

Same road as seen from Google Earth. It is straight and does not make a bend.

Fig. [32]

These images were captured by a mobile phone, but were processed with Hugin. There are some misalignments in the background, that are attributed to the lack of a sophisticated zoom function in the mobile camera, especially when images of far horizon objects are captured. Similar images captured with dSLR, produced far better results.

Fig.[33a] **&** Fig.[33b]

These two images (out of nearly 200 extracted from video), were suited to create a composite. There are many common features between them, and much overlapping so it is possible that many suitable control points will be generated.

Fig. [34]

Preview window of Hugin's simple interface. Since no EXIF metadata were found, FoV was set to a common value of 50.

Fig. [35]

A very strong ghosting (parallax) effect. Ghosting effect occurs when, an object moves between captures, but in this case it seems that the camera changed slightly its perspective.

Fig. [36]

Hugin's advanced interface. Manually introducing six control points. Hugin normally generates 20 to 30 control points to accurately align a series of images, in a time period counted in seconds. Manually introducing these six control points, took nearly two hours.

Fig. [37]

No parallax errors but the images show a very small misalignment.

Fig. [38]

Implementing the SURF algorithm in both images. A very large number of common features were discovered especially in those areas of the images that had either greater light exposure or were resembling geometric forms.

Fig. [39]

Light-field imaging, offers a new approach to digital photography. The focal distance and depth of field, can be altered in post processing after a photo is taken. - Near focus (top), Far focus (middle), Full depth of field (bottom).

Fig. [40]

The plenoptic function describes the information available to an observer at any point in space and time. Shown here are two schematic eyes-which one should consider to have punctate pupils-gathering pencils of light rays. A real observer cannot see the light rays coming from behind, but the plenoptic function does include these rays.

Fig. [41]

The plenoptic function describes, all of the image information visible, from a particular viewing position.

Fig. [42]

The 4D light-field is created by eliminating factors from the plenoptic function. Inside the camera, only two measurements needed, in order to create a 4D light-field. The angle of the ray, as it passes through the main lens, is the first measurement (θ,φ,), and the position of the pixel on the camera secondary microlences is the second (Vx, Vy).

Fig. [43]

A taxonomy of plenoptic functions.

Fig. [44]

Traditional (a) vs. Lightfield camera (b). The majority of cameras that capture 4D light-fields use a micro lens array to modulate the incoming light before it hits the image sensor. Moreover, the sensor uses angle sensitive components that measure the angle of the incoming ray, in addition to its energy.

Fig. [45]

The evolution of light-field cameras. Top the original experimental setups: Stanford Large Statue Scanner (a), Lego Mindstorms Gantry (b). Bottom: Commercial light-field cameras from Lytro and Raytrix.

Fig. [47]

Even when choosing pairs of images out of the first and the last image of a line or a column, in order to maintain maximum separation, they still had nearly 90% overlap, so a small FoV is expected in the final composite.

Fig. [46]

Not only the final composite had a narrow field of view, it also presented a very strong parallax effect.

Fig. [49]

The same procedure as above but this time, these images are extracted from the online viewer [LIR15].

Fig. [48]

FoV is still narrow, but the parallax error is eliminated at least.

Fig. [50]

An experimental setup and the method, for constructing concentric mosaics.

Fig. [51]

Three examples of concentric mosaics.

Fig. [52]

The workflow for creating light-field composites, from focal stacks. All of the methods of light-field panorama imaging that were examined up to now, involve some kind of 4D to 2D transformation, applying known digital panorama imaging techniques, and then reconstructing the light-field. This process is prone to a lot of errors and has many limitations.

Fig. [53]

Eight intrinsic plenoptic camera parameters: wa ,ha ,wf ,hf ,ou ,ov ,duv , and dwx . UV and ST are the focal and perspective planes inside and outside the camera housing. WX is the plane that intersects a predefined reference point on the camera housing. All planes are parallel and perpendicular to the camera's optical axis (dotted line). They all depend on zoom and focus settings of the camera's main optics

Fig. [54]

Two plane light-field parameterization (2PP) is converted to cylindrical: Multiview circular/perspective projection over two nested cylinders ST and UV. The horizontal and vertical perspectives are defined by the angle β and the height h. One ray is parameterized by β, h and the intersection with the UV cylinder (u,v).

Fig. [55]

Registration of two adjacent input light-fields. Choosing a correct set of parameters dr and α, is integrall in registering two adjastent light-fields, without errors. The overlap between the input light-fields should be as large as possible, while the number of recordings should be as small as possible.

Fig. [56]

Re-parameterization and blending of two adjacent input light-fields: Projecting the vertical center line in the V-direction of the UV plane through all horizontal samples in the S-direction on the ST plane of the registered input light-field onto the ST cylinder yields all possible horizontal perspective coordinates β j , while for the vertical perspective hk = tk. All necessary rays of the same input light-field are alpha-blended (transparency) with other input light-fields for the same βj.

Fig. [57]

Geometric relationship between two-plane parameterization for given intrinsic camera parameters and optimal choice of cylindrical parameterization that full light-fields the following constraints: As many of the recorded input rays as possible should be retained without causing empty samples in the resulting panorama light-field. The overlap between the input light-fields should be as large as possible. The number of recordings should be as small as possible.

Fig. [58]

Calibrated intrinsic and optimal recording parameters of Lytro plenoptic camera in everyday mode (default refocus range) for three different zoom steps with Eα =2o and Edr =2mm.

Fig. [59]

Panoramic tripod head: construction drawing (left) and 3D print (right). The adjustable slide allows manual α-rotations and dr-shifts.

Fig. [60]

Comparison of this method with a naive multi-perspective image stitching of input light-fields: Since the spatial and directional domains are processed independently, the resulting panorama light-field is inconsistent in the directional domain. This leads to strong artifacts in rendered images, in particular for refocusing (bottom).

Fig. [61]

Comparison the focal stack approach: The overlaid perspectives of the original input light-fields (top) display the recorded scenery and light effects that should be retained in the resulting panorama light-field (center and bottom). This is the case for this method (center). The focal stack approach (bottom), however, does not preserve correct anisotropic reflections and refractions and causes artifacts at occlusion boundaries. The blue arrows indicate some of these errors. The color coded difference images (right) illustrate the variation between left-most and right-most light-field perspectives.

Fig. [62]

The workflow of this light-field stitching algorithm. Each 4D light-field is presented as a 2D plane for ease of visualization.

Fig. [63]

Each ray [s,t,u,v] from L, is mapped in the refference L′ [s′,t′,u′,v′], with the use of RMSS. By applying the RSMM transformation, two light-fields are registered under a same 2PP coordinate.

Fig. [64]

RSMM transformation errors: Even though these light-fields have large overlapping subspace the rays do not match exactly due to under sampling (left). The scene may have changed a little over time and ghosting artifacts due to motion were introduced (right).

Fig. [65]

How the camera acquires light-fields: (a) The light-field camera is panned and tilted in order to increase the FoV. (b) To enhance synthetic aperture and parallax, light-field camera is translated (white arrow); To enhance rotational parallax, the camera is rotated around the object.

Fig. [66]

A panoramic light-field generated from a 5×4 grid of light-fields. Left: full view focusing at a plane close to the foreground flower. Right 2×2 images: close-up views of highlighted regions focused at the background (top row) and foreground (bottom row).

Fig. [67]

A stitched light-field with increased rotational parallax. The top row shows two views from the stitched light-field. The horizontal strip at the bottom is a composite of the profile along the red line as the object is virtually rotated. The strip is the coherrent, with no errors.

Fig. [68]

Comparisons of light-field rendering using RMSS method vs. the method described in S8.iv, using the same light-field data. RMSS method is able to stitch the input into a ghosting free panoramic light field while the method described in S8.iv produces strong ghosting artifacts due to translational motions across the input light-fields.

Fig. [69]

Failure cases. Top row: the depth-of-field rendering on the in-focus region (the pupil) exhibits blurs due to errors in RSMM estimation. Bottom row: the flower is incorrectly stitched due to large displacement between the two light-fields.n/wfarad1846.html