



DIY Camera Array 2: Computational Refocusing With Just One Camera
by [daniel_reetz](#) on December 18, 2009

Table of Contents

License: Attribution Non-commercial Share Alike (by-nc-sa) 2

Intro: DIY Camera Array 2: Computational Refocusing With Just One Camera 2

step 1: Oh Man Here Comes Some Theory; Stick With Us. 4

step 2: Capturing the damn images already. 7

step 3: Refocusing with LTextures. 9

Related Instructables 11

Advertisements 11

Comments 11

License: Attribution Non-commercial Share Alike (by-nc-sa)   

Intro: **DIY Camera Array 2: Computational Refocusing With Just One Camera**

DIY Camera Array 1: Computational Photography Primer.

DIY Camera Array 2: Computational Refocusing With Just One Camera.

This first Instructable is a primer on the field of Computational Photography, which is a new field of research that is developing extremely powerful cameras. These cameras allow the Depth of Field, the object in focus, and the position of the camera to be modified *after the picture is taken*. None of those things are possible with a traditional camera.

We show how to build one kind of computational camera -- a light field array. There are many other designs out there.

Before you go to all the trouble of building your own array, you should know that you can play with computational refocusing with just one camera.



You can think of it like this: 12 cameras taking a picture in a row is the same as 1 camera taking 12 pictures in a row.



Can work like this:



What you need:

A digital camera. (Any kind is fine)

A clamp.

A ruler, T-square, or meterstick.

A table or flat surface.

Patience.

Time.

LFTEXTURES.



step 1: Oh Man Here Comes Some Theory; Stick With Us.

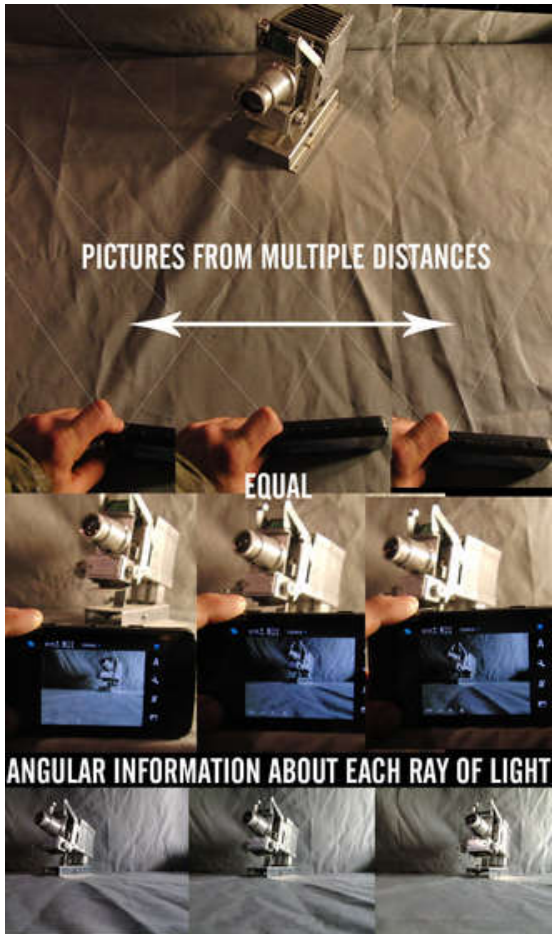
There are many methods to achieve computational refocusing. **We do it the simplest way.**

I'm going to try to explain this intuitively and imprecisely, because the mathematical definition has been well explained elsewhere.. First of all, light field imaging treats light as rays, each with a direction and intensity. So think of the light around you as a bunch of long thin rays traveling through space. They shoot around in all directions, all the time, reflecting off things, refracting through glass, air, water, etc.

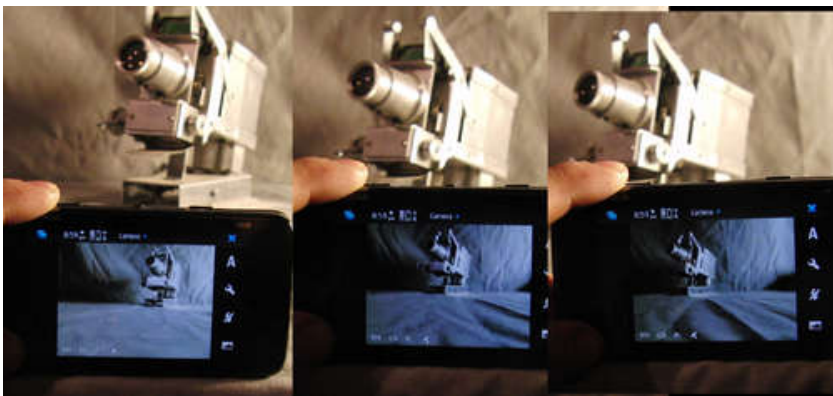
What we want to do is select **certain** light rays from **certain** angles to form our final image. To do this, first we need to actually capture rays which are coming in from different angles.

It goes without saying that the most obvious way to capture rays from different angles is to measure them from different points in space. What is not so obvious is how to recombine these intermediate images into the final, refocused image.

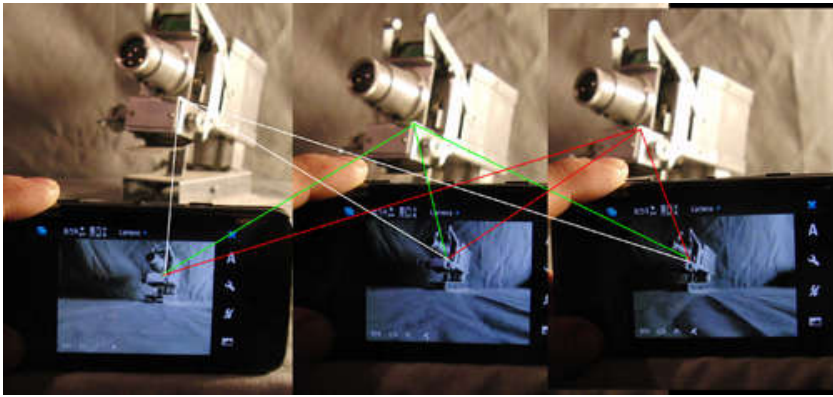
Let's try to think of this **in terms of cameras**. Imagine you have an object sitting on a table. You have a camera centered on it. You take a picture.



When you move the camera left and right, the object moves right and left in the image, respectively.

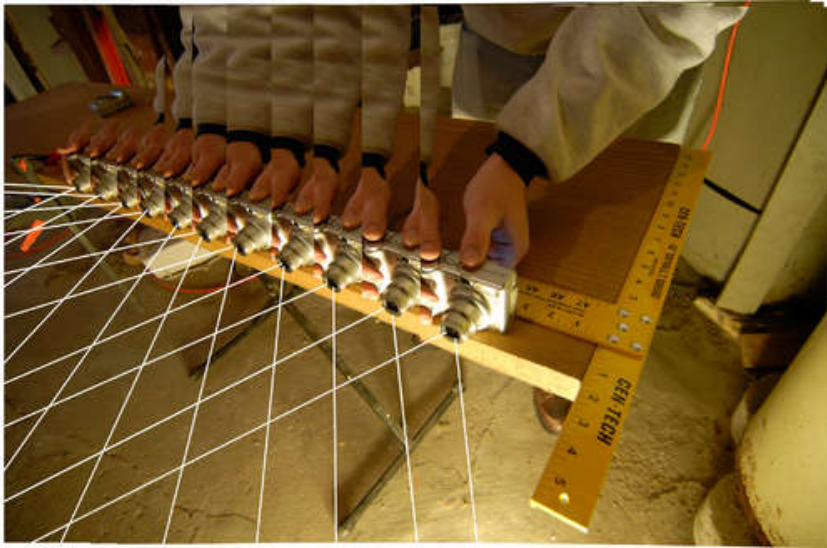


Think about this for a minute -- by looking at the object like this, you have captured different rays of light moving in different directions from the surface of the object. And in the extremes of the left and right image, you can actually see around the object a little bit. If you took enough pictures, you could "see through" the object by choosing only the rays of interest.

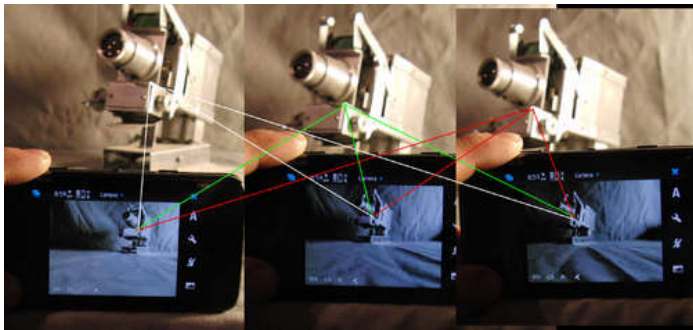


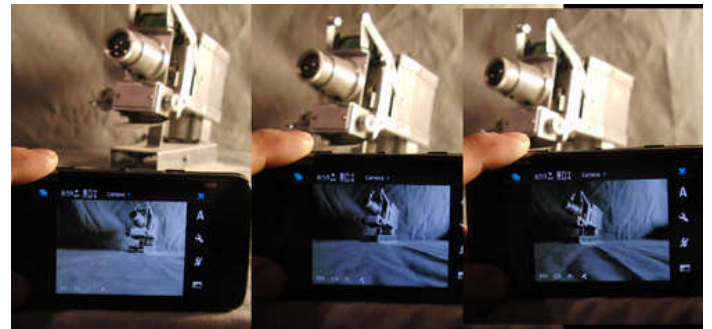
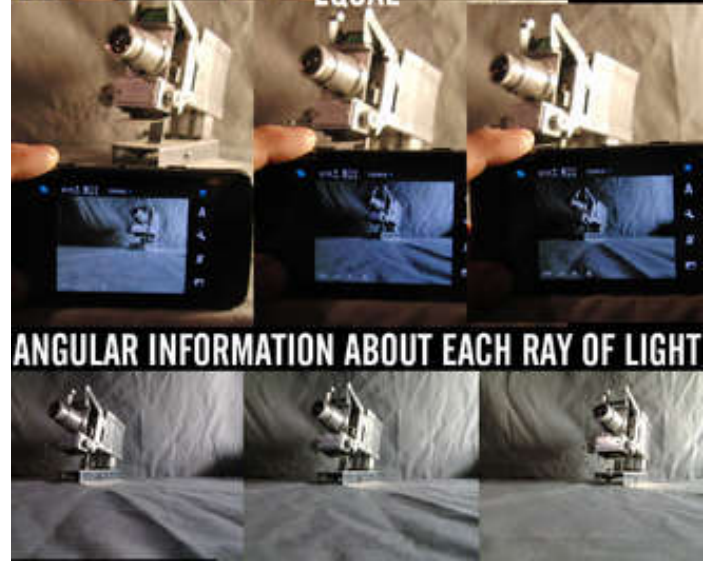
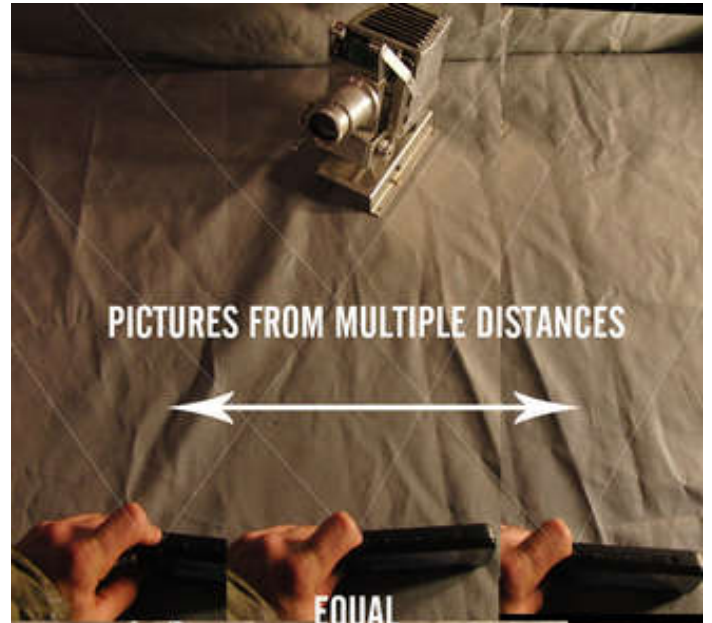
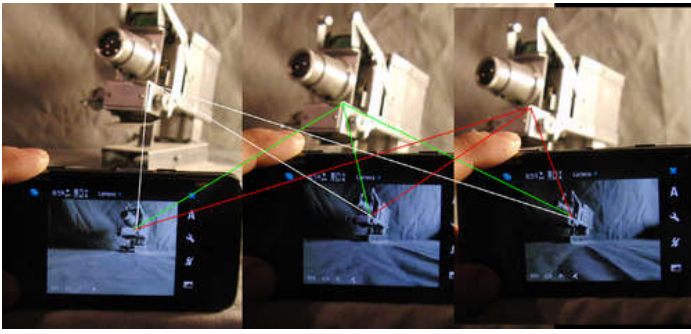
The clever trick here is that by placing our cameras at exactly the same distance apart every time, we "know" the angle of incoming rays relative to each other. We don't know them in an absolute sense (like in radians or degrees), but that's not really very important -- if we overlay the images and shift them all relative to each other, we "select" different rays by blurring out the ones we're not interested in.

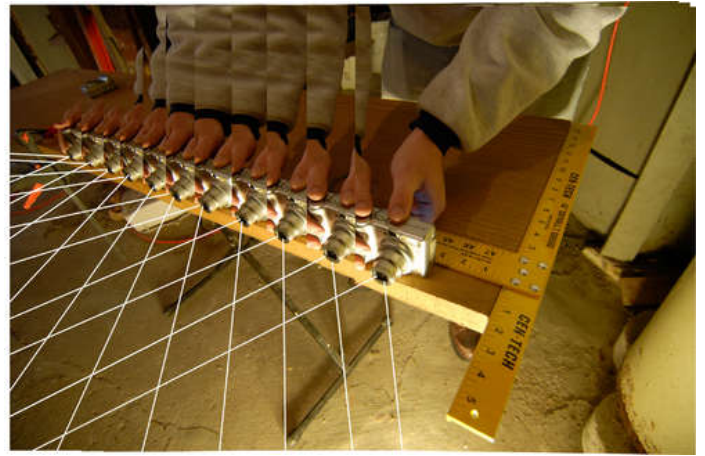
In other words, the important thing is that by making the distance between the cameras the same every time, we automatically preserve the angular relationships between incoming rays.



?







step 2: Capturing the damn images already.

So how to do that? Well, it's simple.

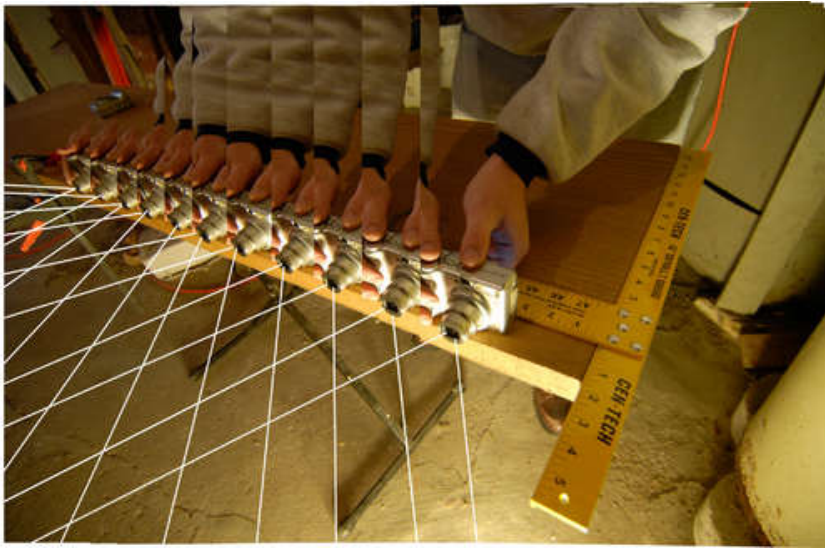
Just clamp a ruler, t-square, or meter stick to a table.



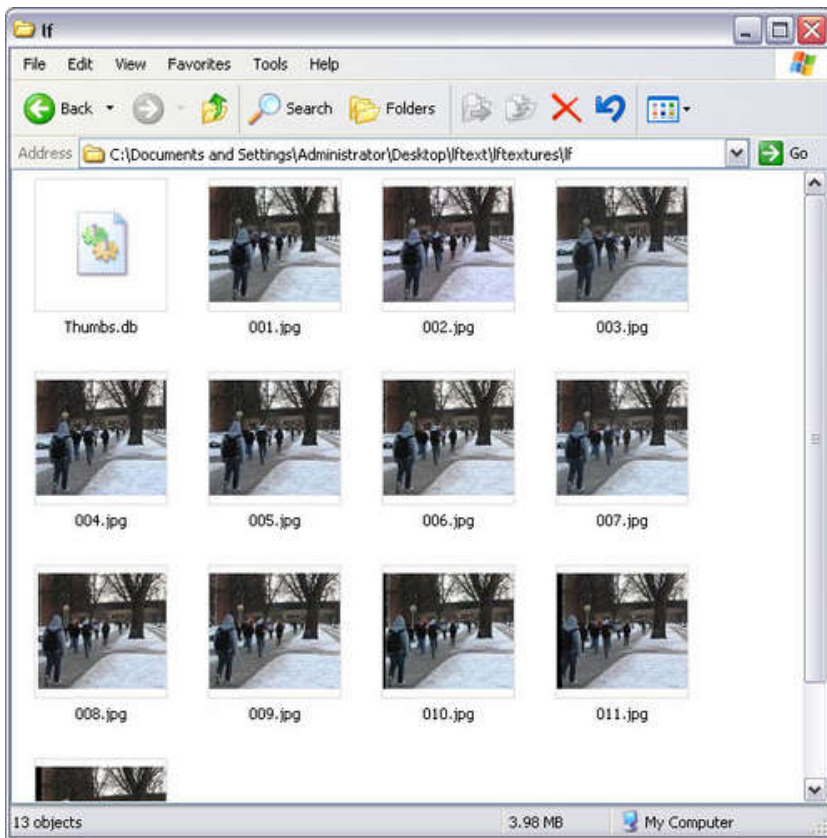
?
Set your camera on the table, pointed at whatever you are interested in photographing.

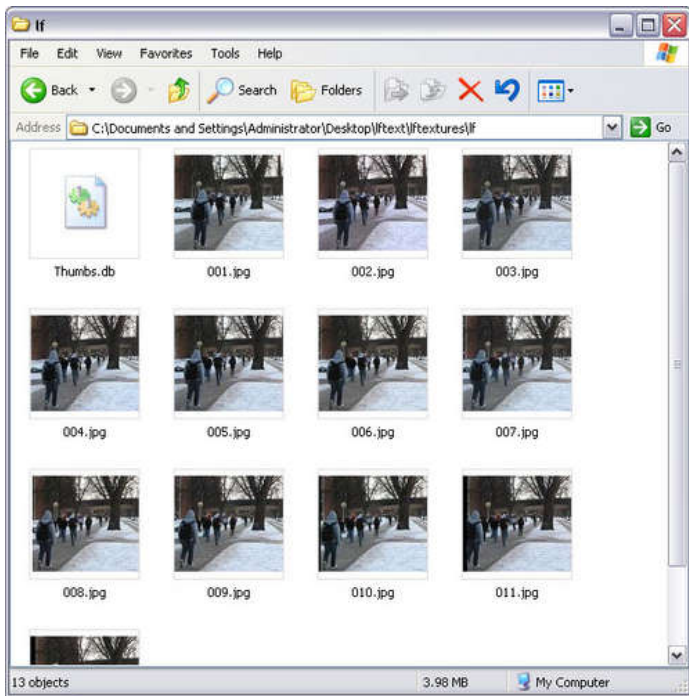
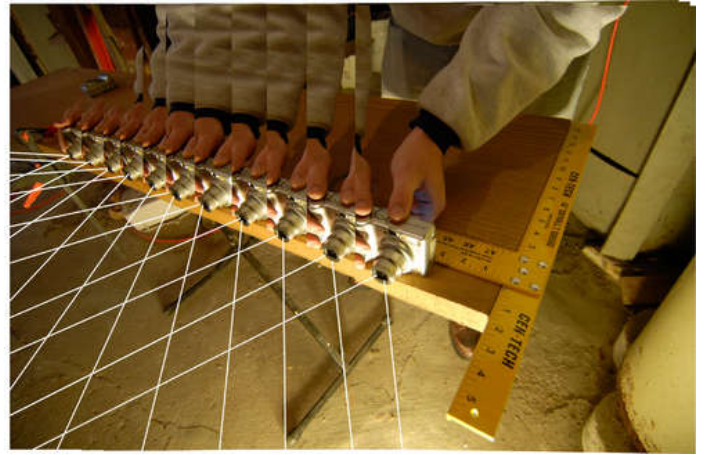


Move the camera from left to right or right to left, moving exactly the same distance every time and take a picture at a some interval -- every 2 inches is a good start.



Copy this set of images to a folder on your computer.





step 3: Refocusing with Lftextures.

This is where Lftextures comes in. Matti Kariluoma wrote Lftextures in C with OpenGL. The primary purpose of Lftextures is to allow you to easily do this refocusing.

So go download it already. :) To install it, just unzip the contents of the folder. To start it, just click on "lftextures.exe" and wait.

To refocus, Lftextures makes all the images transparent, and lays them on top of each other. By scrolling the mouse wheel, you are able to refocus your captured image by shifting the images relative to each other. By pressing "P" you save a copy of your refocused image in the working directory of lftextures.

We provide a couple of data sets for you to play with, but we hope you'll go try capturing your own. To use them, just copy your images into the "lf" directory that comes in the Lftextures archive. Be sure to delete the ones that are there already.

Please remember that Lftextures only works if your images are perfectly evenly spaced. If you get nothing else from this Instructable, please get that your images must be on a single line and perfectly spaced!!!



That's it for the moment. The next few articles will be about how to build the array we built, which makes a lot more interesting things possible.



Related Instructables



Laser Triggered High-Speed Photography by SaskView



Remote for Canon Compact Cameras by j44



High power flash. (slideshow) by VadimS



Better closeups by neelandan



Filter holders for point-and-shoot cameras by tintinnuntius



Hacking the Walgreens Forever camera for near free film by mpap89

Advertisements

Comments

[10 comments](#) [Add Comment](#)



VadimS says:

Thought I should mention it's worth using a camera capable of setting everything manual, or using CHDK on a Canon so that the images are consistent. Same exposure, white balance etc.

Oct 19, 2010. 10:42 AM [REPLY](#)



Uncle Kudzu says:

it would be interesting to see some aerial computational photography.

Dec 21, 2009. 8:47 PM [REPLY](#)

i have seen overlapped stereo flight lines (large format conventional photography) that yielded some information for mapping purposes, and it seems like something similar done with your camera array might be useful for that sort of thing.



PS118 says:

Great stuff, and 1000 bonus points for the free software!

Dec 21, 2009. 4:11 PM [REPLY](#)

I've seen some of the research before, and I seem to recall one sensor with multiple lenses.

Since then I've wondered about those bug-eye viewers kids play with. Granted they're cheap plastic, but I wonder if you could get any worthwhile data if you fitted one to a camera??

Example:

<http://www.physlink.com/estore/cart/FlyEyeViewer.cfm>



Uncle Kudzu says:

Dec 19, 2009. 11:17 AM [REPLY](#)

hmmm... so every single thing in the photograph(?) is in focus in some way? looks like a shifting depth-of-field in the animation.

any use beyond being very interesting? could you make a single image with infinite depth-of-field, for instance?

thanks for sharing this!



mikeasaurus says:

Dec 19, 2009. 11:14 PM [REPLY](#)

"could you make a single image with infinite depth-of-field, for instance?"

Couldn't you achieve this by keeping the camera stationary and just taking photographs as you refocus, then import to PS and compile the image?



daniel_reetz says:

Dec 20, 2009. 3:21 AM [REPLY](#)

Yes, you could. But a normal camera couldn't "see through" an entire person, especially a tiny little compact camera. This camera has abilities that far exceed each little camera that comprise it.



mikeasaurus says:

Dec 20, 2009. 9:43 AM [REPLY](#)

Judging from the images you've shown here I'm not "seeing through" anything. As **Uncle Kudzu** mentions, is there a practical application for this? If you're attempting to see through, or see around someone wouldn't it be easier to just use 2 cameras set apart to create a stereoscopic image?

Maybe in the intro you could go into some detail about what the aim of this technique is, and the advantages and disadvantages.



daniel_reetz says:

Dec 20, 2009. 10:39 AM [REPLY](#)

If you can't see how computational refocusing "sees through" things, I suggest you look a little closer. The same principle is applied every day in synthetic aperture radar and works just fine. The fact that our array has artifacts is a result of the density of the array, which we explained.

You seem to be missing the primary advantage, which is that this is all achieved computationally after the picture is taken. If you don't see why that's useful, think of the many times in photographic history that people have taken photos with the "wrong" subject in focus.

This method captures and exploits angular information and allows lots of control in post as a result. Depth of field, the object in focus, and position of the virtual camera are all able to be somewhat modified after the picture is taken. None of those things can be done with a traditional camera.

If you have a deeper questions about light field photography, have difficulty with the concept of a camera that captures more than a flat image with all parameters baked in, or still wonder why you might want some image control after the moment of capture, I suggest you start reading the papers of the authors I linked in the first Instructable. For more sophisticated results, check out the work of Todor Georgiev.



mikeasaurus says:

Dec 20, 2009. 1:01 PM [REPLY](#)

"This method captures and exploits angular information and allows lots of control in post as a result. Depth of field, the object in focus, and position of the virtual camera are all able to be somewhat modified after the picture is taken. None of those things can be done with a traditional camera."

This is what should be included in your intro.



daniel_reetz says:

Dec 21, 2009. 2:11 AM [REPLY](#)

I edited the intro according to your suggestion. If you have any other ideas, let me know.