Computational Imaging Report

UC2 FPM, DiffuserCam, and Speckle Imaging

REU: Christina Dong, Erin Obermayer, Samuel Perales, Xingzi Xu
HMC: Alec Vercruysse, Rosey Sams, Shreya Sanghai
Advisor: Joshua Brake

May - July 2020

Contents

1 Introduction

Conventional lens imaging systems are heavily integrated in our day-to-day lives. These imaging systems are also deeply important to biomedical fields for the ability to image through biological tissue. While the uses and applications are diverse, these imaging devices all rely on lenses. Thus, the pursuit of improving imaging systems is often in tandem with producing better or bigger lenses. However, this makes high quality imaging systems costly and larger, which ultimately limits accessibility. With modern advancements in computers, it is possible to utilize the computer to overcome the issue of accessibility with lens imaging systems.

Previous advancements in the field of computational imaging including Fourier ptychographic microscopy (FPM) by Zheng, et. al. in 2013 [1]; the UC2 system by Diederich, et. al. in 2020 [2]; single-exposure 3D imaging by Antipa, Kuo, et. al. in 2018 [3]; and speckle imaging by Katz, et. al. in 2014 [4] have made our exploration into these computational imaging techniques possible. In this report, we describe and implement three different computational imaging systems: UC2 FPM, DiffuserCam, and Speckle Imaging.

The goal of the UC2 FPM project is to achieve high space-bandwidth product (SBP) and field-of-view (FOV) in low-quality lens imaging systems by merging the development of the UC2 System with FPM. The SBP is a metric which measures the resolution of an image, and the FOV quantifies the size of the image. The UC2 System is an open-source, low-cost optical toolbox that has been proven to be customizable and versatile, and characterized to be what Arduino is for electronics [2]. FPM is a computational technique that allows us to obtain higher resolution images. Together, UC2 and FPM create an accessible, low-cost, and high-quality microscopy system.

Another way to keep imaging systems at a low cost is to remove the lens. DiffuserCam works to construct 3D images from a single, lensless, 2D data measurement [3]. Speckle imaging works to non-invasively image through scattering mediums, such as biological tissues, by using known properties of the scattering medium in place of a lens [4]. Prior to the development of computers, lenses were the preferred method of imaging objects due to its simple transfer function. However, if the transfer function of a random object is known, said random object can replace the lens in an imaging system, and we can utilize a computer to handle its complex transfer function. Both the DiffuserCam and speckle imaging exploit this idea to image objects.

The goal of these projects was to show proof of concept for each computational imaging system, which included working with both hardware and software. We were able to successfully build the physical systems and have reached the stage of troubleshooting and optimizing each system. We also implemented the necessary software for image processing, but recognize that the processed images' quality can still be improved.

This report will first cover several topics to aid in the understanding of computational imaging systems: Fourier optics, phase retrieval, point spread functions, convolution and autocorrelation. In the Methods section, we go over the theory, hardware, and software for each project. We then will discuss the progress and results we found and made in each project in Results. Following this, we talk about how each project can be continued in Potential Future Work.

2 Background

2.1 Fourier Optics

The Fourier transform has been a powerful mathematical tool used for centuries to deconstruct an input signal into the amplitudes and frequencies of the sine waves that compose it. Typically, one might consider input waves to be electrical signals or sound waves, but the foundation of a large portion of modern day optics is considering the 2-dimensional Fourier transform of an input image. The use of Fourier transform is quite natural since one of the properties of a traditional lens is that the pair of focal planes of the lens are Fourier conjugate. This means that a lens can actually be used to take a Fourier transform of some input image.

Each point of the Fourier transform spectrum represents a two-dimensional sine wave's direction, frequency, and amplitude, such that when all of the sine waves are added together, we get the original image. Once we know the spatial frequencies of an input image, we can filter and manipulate them such that performing an inverse Fourier transform will result in some desired outcome. An example of this is shown in Fig. 1.

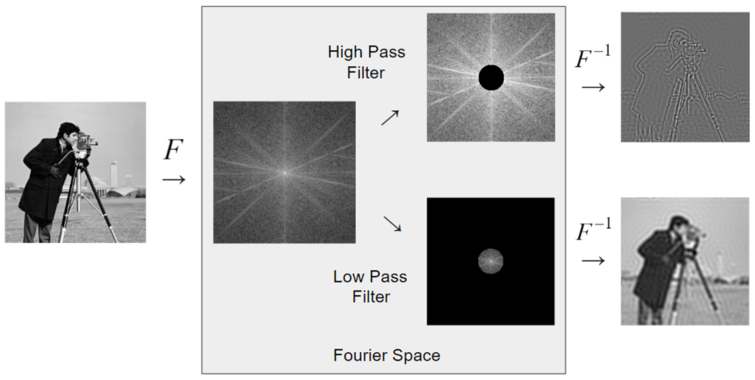


Fig. 1: By filtering out lower frequencies (High Pass Filter) and inverting the Fourier transform, we get edge detection. By filtering out higher frequencies (Low Pass Filter) and inverting, we get image compression.

A common setup in optics in the $4f$ System. This is a setup in which two lenses are placed at each others focal length. The first lens acts as a "beam expander" by taking some input light and collimating it. The light then passes through some transparent image and into the second lens. This second lens then takes the Fourier Transform of the image and outputs it. This system is an easy way to capture the Fourier Transform of some sample.

Each project utilizes the Fourier transform in some capacity; it can be used in many different algorithms to recover information that initially appears to be missing from our captured image data. The $scipy.fftpack$ module in Python can simulate FT and IFT in one or two dimensions.

2.2 Phase Retrieval

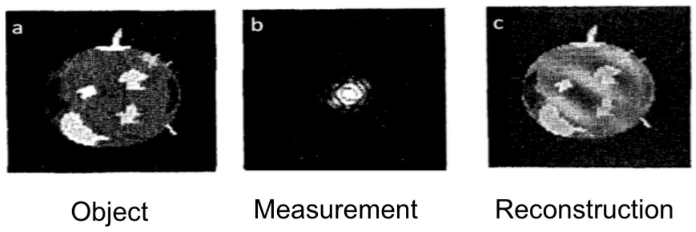


Fig. 2: Without phase information of an object, cameras can't capture a complete measurement of that object. A reconstruction using phase retrieval algorithms must be performed to get a better image. Image from Fienup's 1978 paper [5].

Light waves are composed of a magnitude and phase. However, light waves oscillate so quickly that most electronics cannot measure the phase information of the wave. In Fig. 2, the discrepancy between the object and measurement images shows the importance of phase information to an accurate image. Thus, phase retrieval algorithms are used to recover the lost phase information to produce better images. The iterative process of phase retrieval algorithms is shown in Fig. 3 below.

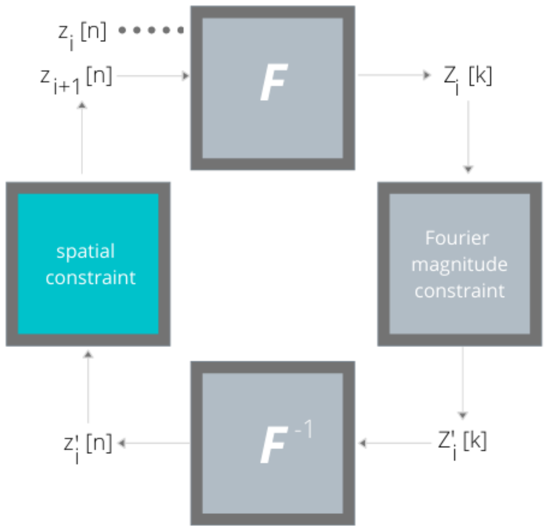


Fig. 3: General outline of phase retrieval algorithms. n denotes spatial coordinates, and k denotes Fourier coordinates.

Most phase retrieval algorithms follow the steps:

1. Initialize algorithm by defining the light wave equation using either a measured or approximated spatial magnitude along with a spatial phase guess, which gives $z_i[n]$.
2. Fourier transform $z_i[n]$ to obtain $Z_i[k]$.
3. Impose Fourier magnitude constraints but keep Fourier phase the same to obtain $Z_i^{'}[k]$.
4. Inverse Fourier transform $Z_i^{'}[k]$ to obtain $z_i^{'}[n]$.
5. Impose any spatial constraints to obtain $z_{i+1}^{'}[n]$. This will be fed into the next loop of the algorithm.

Many different phase retrieval algorithms follow this outline and differ only in step 5 or the block highlighted in blue in Fig. 3. For instance, in the Gerchberg-Saxton (GS) algorithm, the spatial constraint would be the spatial magnitude, necessitating the knowledge of both the object's Fourier and spatial magnitude in order to retrieve phase. The GS algorithm would work for two measurement imaging systems, with the two measurements being the spatial and Fourier magnitude.

There also exists algorithms for single measurement imaging systems, with the single measurement being the Fourier magnitude. For example, the hybrid-input output (HIO) method is one such algorithm and uses the following constraint:

$$z_{i+1}[n] = \begin{cases} z_i'[n], & n \notin \gamma \\ z_i[n] - \beta z_i'[n], & n \in \gamma \end{cases}$$

with β being a parameter close to unity (typically between 0 and 2), and γ being the set of conditions for which $z_i'[n]$ violates the real spatial constraints. These constraints are typically realness and positivity.

Another single-measurement, phase-retrieval algorithm is the error reduction (ER) method. ER uses the following constraint:

$$z_{i+1}[n] = \begin{cases} z_i'[n], & n \notin \gamma \\ 0, & n \in \gamma \end{cases}$$

with γ having the same definition as in the HIO method.

Both the HIO and ER method are used in the speckle imaging project to reconstruct images. Furthermore, the idea of using algorithms in image reconstruction is prevalent in all three projects. There are many other algorithms such as Input-Output method, Output-Output method, OSS algorithm, etc. For more detail on these methods, see [5] and [6].

2.3 Point Spread Functions

A traditional imaging system usually has a point-to-point mapping of light; a single input source of light will result in a point source output. In many cases of computational imaging, however, an input point source of light will output a complex pattern, due to the function of the system and how light is refracted. These outputs, called point spread functions (PSFs), can be used to describe the transfer function of an optical system, see Fig. 4.

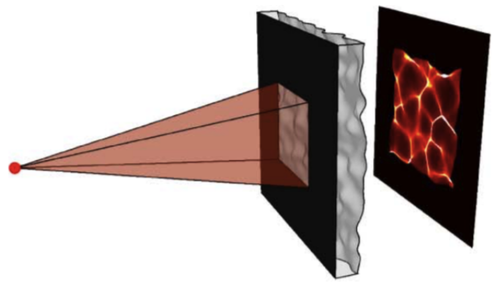


Fig. 4: As a point source of light is sent through the DiffuserCam, the output (PSF) is a unique caustic pattern. Image from Antipa's 2018 paper [3].

Lensless imaging has sparked a strong interest in the field of optics because of its accessibility. These systems replace a standard lens with a diffuser or other unconventional materials with varying textures and opaqueness. When light is sent through these materials, it refracts light differently depending on the surface creating unique point spread functions. The information encoded in these PSFs along with algorithms can then be used for image reconstruction.

Shift invariance refers to the behavior of PSFs. If an input point source is moved laterally or axially, the pattern that is created should remain the same pattern, only shifted with respect to the point source. Fig. 5 shows the shift invariance of the DiffuserCam system. A system is defined as linear shift invariant if it demonstrates that the output intensity is directly proportional to the input intensity. Therefore, if the input intensity is doubled, the output is doubled as well.

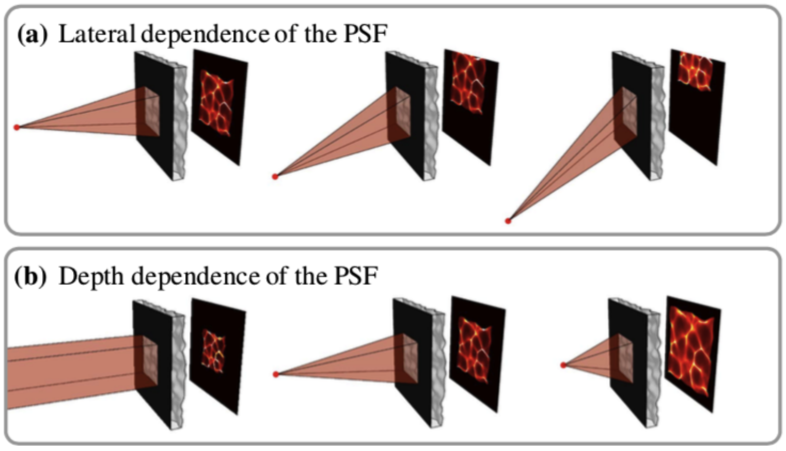

(a) Lateral dependence of the PSF
(b) Depth dependence of the PSF

Fig. 5: Shift invariance of the DiffuserCam. When an input point source is shifted laterally (a) or axially (b), the output PSF shifts accordingly. Image from Antipa's 2018 paper [3].

2.4 Convolution & Autocorrelation

Convolution

Convolution is widely used in signal processing. Electrical engineers have created this scheme where we can describe a system and an input signal with separate and different mathematical equations. Under this scheme, the output signal is the convolution of the system function and the input signal function. Similarly to how FT was adapted from 1-D signals to 2-D images, convolution can also be used in 2-D images. Fourier optics is a good example of this, since the output image we see is the convolution between the input image and the imaging system.

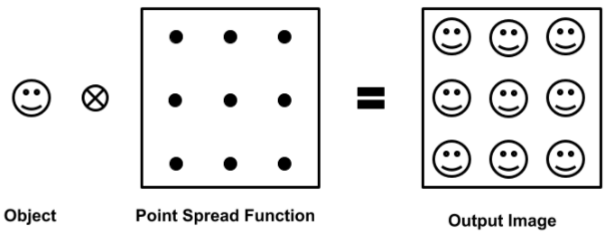


Fig. 6: Visualization of convolution: output image is formed by convolving the PSF with the input object image.

Mathematically, a one dimensional discrete convolution is described as:

$$y[n] = x[n] * h[n] = \sum_{l=-\infty}^{\infty} x[l] \cdot h[n-l].$$

Here, the $*$ sign indicates convolution, $x[n]$ is the input signal, $h[n]$ is the system transfer function, and $y[n]$ is the output signal. n is the indexing variable, and l is the dummy variable.

All three projects are heavily involved with convolution, as it gives a mathematical way to model our imaging systems. Specifically, when simulating convolution with Python, we can use the $scipy.signal.convolve$ module.

We can also simulate convolution with FFT (fast Fourier transform) and IFFT (inverse fast Fourier transform) methods:

$$conv(g,h) = IFFT\{FFT\{g\} \cdot FFT\{h\}\}$$

.

We have created a convolution demo in Google Colab for future students to better understand the mathematical operation. This demo covers 1D manual convolution, convolution using the imported Python numpy library, 2D convolution, and convolution with the Fourier Transform. The notebook can be found here: `https://colab.research.google.com/drive/1VBZz96F-bM80jlkdtUnh7PZaKCivpjXm?usp=sharing`.

Autocorrelation

Autocorrelation describes the similarity a signal has with itself over a certain dimension. For instance, we expect the temperature tomorrow be more similar to the temperature today than the temperature two months later. Thus, if we take the autocorrelation of the temperatures over the time dimension, we will have a higher autocorrelation value tomorrow than two months later.

Mathematically, a one dimensional autocorrelation can be described with convolution. Specifically:

$$c_{xx}[n] = x[-n] * x[n] = \sum_{l=-\infty}^{\infty} x[-k]x[n-k].$$

Here, $c_{xx}[n]$ is the autocorrelation, $x[n]$ is the function whose autocorrelation is calculated, and the symbol $*$ indicates convolution. Again, n is the indexing variable, and k is the dummy variable.

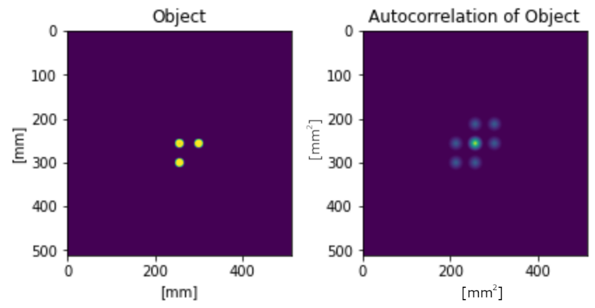


Fig. 7: Example shows relationship between object and its autocorrelation.

The most effective way to simulate autocorrelation in Python is also using FFT and IFFT. The steps is as follows:

1. Fourier transform function.
2. Multiply the result of Step 1 by its conjugate.
3. Take the inverse Fourier transform of the result of Step 2.

Autocorrelation is at the core of the speckle imaging project, where autocorrelation is used to get rid of the random speckle pattern to aid in the image reconstruction process.

3 Methods

Raspberry Pi and Pi Camera

Either to decrease cost or to create an open source research environment, all three projects have chosen to use a Raspberry Pi controlled Raspberry camera. A Raspberry Pi is a mini computer running an OS of its own. It uses an SD card for memory storage, and it outputs HDMI signals to a monitor. A very useful website for debugging problems in setting up a Raspberry Pi is [7]. The Raspberry Pi camera captures an accumulation of light. As shown in Fig. 8, saturated pixels can damage the speckle patterns containing crucial information. Thus, it is very important to make sure that camera pixels don't get saturated via control over exposure time. ISO is also an important parameter to look at, as it specifies whether the camera is capturing image in a dark or bright environment.

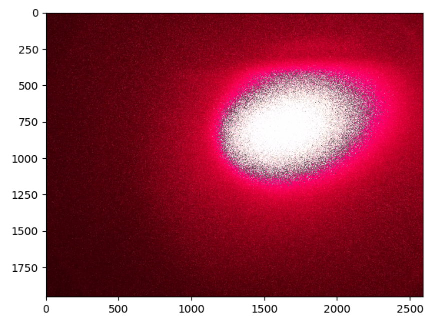


Fig. 8: Saturated images can damage crucial speckle pattern information. Units in px.

3.1 UC2 FPM

Theory

Fourier Ptychographic Microscopy (FPM) [1] is a computational imaging method in which, by taking images of a sample from varying illumination angles at a high FOV, we capture information about different parts of the sample's Fourier spectrum. Using an iterative algorithm, we can 'stitch' these various images together in Fourier space after which an inverse Fourier Transform will yield a higher resolution image. The goal of UC2 FPM is to implement this algorithm with the low-cost, 3d-printed, and modular UC2 open-standard system.

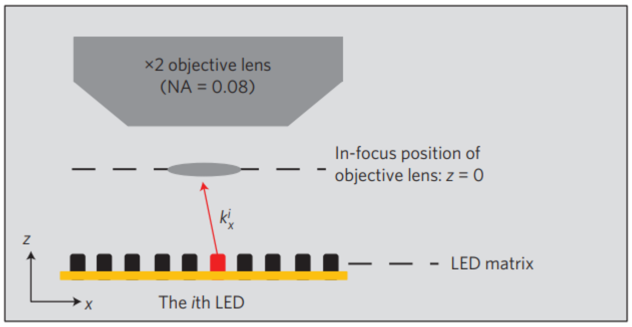


Fig. 9: Image acquisition consists of illuminating a sample from the N^2 different LEDs on an N x N matrix. Images taken with LEDs at a larger angle with the sample must have a longer exposure time to correct for lower amounts of light reaching the sensor. Image from FPM paper [1].

The procedure of FPM after image acquisition is as follows:

1. Estimate a high-resolution reconstruction of the sample as an up-sampled acquired image and take the Fourier Transform of this estimate.
2. Select a subregion of the estimated image spectrum and take its Inverse Fourier transform to get a low-resolution image estimate for that subregion. This subregion will correspond to an LED used to take an image.
3. Compare the estimated low-resolution image with the corresponding acquired image and update the estimate accordingly.
4. Fourier transform the updated low-resolution image estimate and update the spectrum subregion of the high-resolution estimate.
5. Repeat this process of updating various subregions for all acquired images.
6. Inverse Fourier transform the fully updated spectrum to recover a high-resolution image of the sample with phase information.

Hardware

UC2 [8] defines several low-cost "modules" that can be assembled to make an imaging system. It also includes the hardware required to align the modules on a baseplate. UC2 consists of 50mm unit cubes in which different inserts can be added to interface with different components. Although UC2 is designed to be a *4f-system*, an infinity-corrected system (meaning that the system that has a focal point at infinity) can be set up such that the length between the Microscope Objective (MO) and tube lens can be varied while still keeping the image focused on the sensor plane.

The setup consists of the `ASSEMBLY_CUBE_LED_Matrix_v2` and `ASSEMBLY_CUBE_Z-STAGE_v2` modules to house the LED array and sample respectively, as well as a custom camera module designed to incorporate the Raspberry Pi High Quality Camera, which was not yet supported by UC2. All modules which contained electronics used the MQTT API in order to communicate, conforming to UC2's MQTT specification.

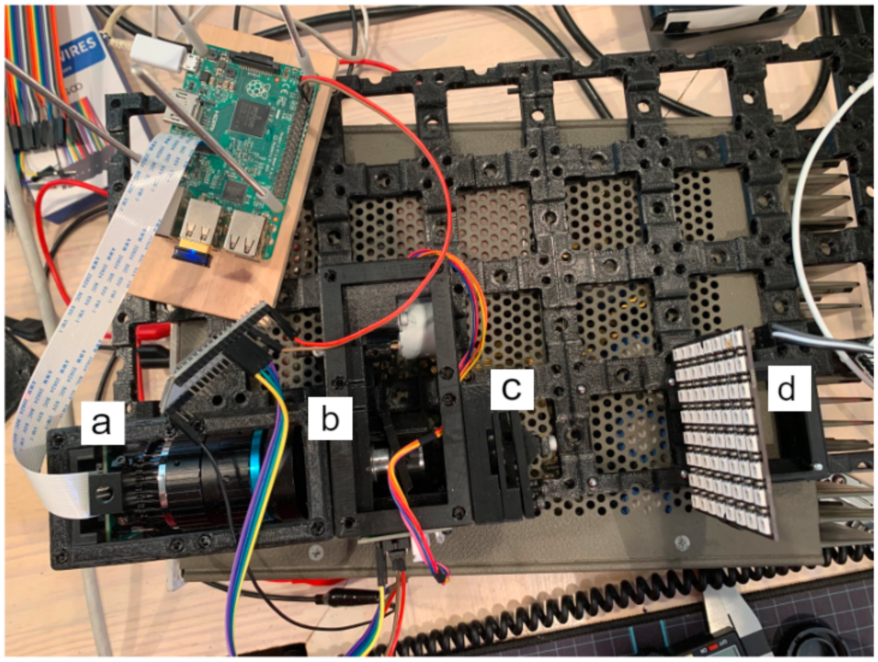


Fig. 10: UC2 Setup. a) HQ RPI Camera, b) Microscope Objective, c) Z-Stage, d) Adafruit NeoPixel 8x8 LED Array

The custom High Quality camera module was designed in order to interface with the new Raspberry Pi HQ camera. The original design used a 2u (100mm x 50mm) cube, with a custom insert designed for both the camera board as well as a 16mm machine vision lens. The machine vision lens was later taken off in the final design in order to increase the magnification (therefore turning the setup to a finite-corrected system, a system with a finite focal point). A Raspberry Pi was used to take images and also interface with the modules containing the LED matrix and Z-stage stepper motor. The camera uses a Bayer color filter array in order to capture color information but because our images are taken under red light, this causes the images to be very sparse. Due to this limitation, in order to sample at the diffraction-limited Nyquist frequency $f_n = 1/2\Delta t$ of the setup, the spatial frequency of the pixels on the sensor had to be double the Nyquist frequency.

The `ASSEMBLY_CUBE_LED_Matrix_v2` module was designed to hold the LED matrix used for the system. It was slightly adapted from the original specification in order to support the mounting hole locations for the matrix we used, an Adafruit NeoPixel 8x8 NeoMatrix. The `WS2812B` LEDs that the matrix uses contain a diffuser in order to provide a visually appealing mix of colors when multiple color channels were turned on. In practice, this meant that the LEDs were more likely to violate the point source assumption that is made in the calculations of FPM. For this reason, a "mask" was

3D printed which included 2mm diameter holes through which each LED could shine. An ESP32 subscribed to a MQTT broker on the Raspberry Pi was used to interface with the LEDS.

The `ASSEMBLY_CUBE_Z-STAGE_v2` module was used in order to be able to translate the MO in the Z-direction (along the optical axis of the system) in order to provide fine-tuned control over focusing. It used a 'lead-screw' style design, using an M3 screw and nut, to convert the rotation of the low-cost `28BJ-48` stepper motors into linear motion, and a linear flexure bearing-based design to allow for the entire module to be 3D printed, other than electronics, screws, and nuts. Another ESP32 was used to interface with the stepper motor driver.

Software

We implement an iteration of FPM which includes Embedded Pupil Recovery (EPRY) [9] so that we can recover information about the aberrations within our system and improve the overall reconstruction. The 'pupil' of the system is the function that the objective lens performs on the image passing through it. The pupil can often have some minor aberrations which cause artefacts in the reconstruction. By estimating this function with EPRY, we can take the aberrations into account.

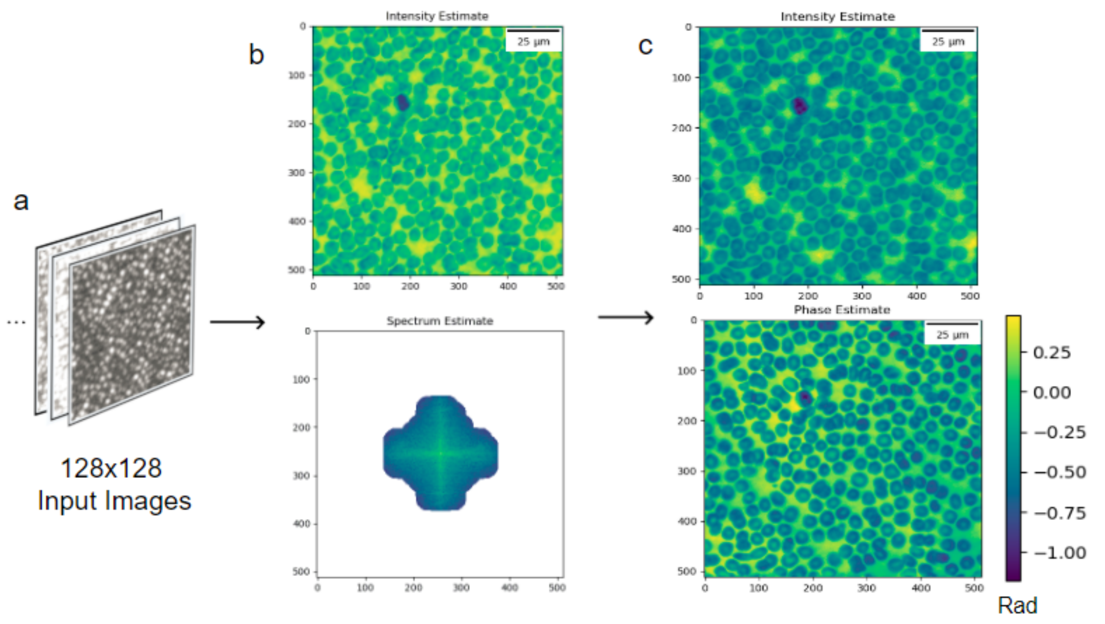


Fig. 11: FPM on simulated blood smear data. a) 64 Low-resolution acquired images. b) A later iteration of FPM showing the current high-resolution estimate along with its estimated spectrum. c) Fully recovered high-resolution image along with its phase (Note that these recovered images are 4x the size of the input images). Images from advisor Josh Brake.

Because of the limitations of the physical system, images require some processing prior to being input into the algorithm. There are three primary parts of processing: demosaicing, denoising, and image segmentation.

Demosaicing: The Raspberry Pi HQ Camera comes included with a bayer filter (BGGR specifically). This essentially designates each pixel a specific color channel (red, green, or blue) as opposed to having each pixel be three channels. Because we take our images under red light only (using multiple wavelengths renders image / phase recovery extremely difficult) we are left with a very sparse red-channel image.

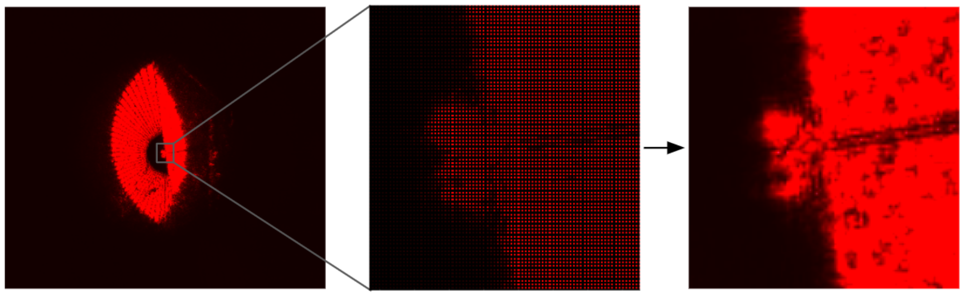


Fig. 12: An acquired image (left) of a transparency sample can be seen in MATLAB as sparse (middle). Using the built in *demosaic* function, we can interpolate the missing data in the image (right).

Denoising: We try to minimize external light passing through the sample by placing a mask on the objective and taking images at night but inevitably, some noise will end up on the sensor. Denoising had already been implemented in the open-source MATLAB code provided by Tian Labs so all this required was adjusting some of the parameters. Noise is estimated for each acquired images by looking at multiple subregions of the image which correspond to the background and averaging the values in these regions together. This average value is then compared to some threshold to see whether this is truly noise or an important part of the image. In the case that it is determined to be noise, we subtract this value from each pixel to essentially cancel out the background of the image.

Image Segmentation: Running simulations on EPRY-FPM is not computationally intensive as the input images are simulated to be relatively small for efficiency (128x128). In practice, the Raspberry Pi HQ Camera captures images of size 4056x3040. 64 images of this size from each of the LEDs on the array makes running the algorithm highly inefficient and requires that we optimize. To do this, each image is segmented into n by n subregions which the algorithm runs on separately. Each image is then stitched back together to recover the full image. This improves the speed of the algorithm but suffers from imperfect recovery as the stitches between each subregion can be highly visible.

To solve this, we pad each of the subregions by some amount such that they overlap with the adjacent subregions. This decreases the contrast between the reconstructions and preserves the improved recovery time. The amount of padding is restricted by how small the subregions are but we've found that around 10-20 pixels of padding is the most useful. Fig. 13 shows figures with and without padding.

Image segmentation also allows for the recovery of the spatially varying pupil functions which is very useful since an aberration may be different from various angles. This means we get both a faster and higher quality reconstruction after a sufficient amount of padding is included. Fig. 14 shows the relationship between recovered pupils and padding.

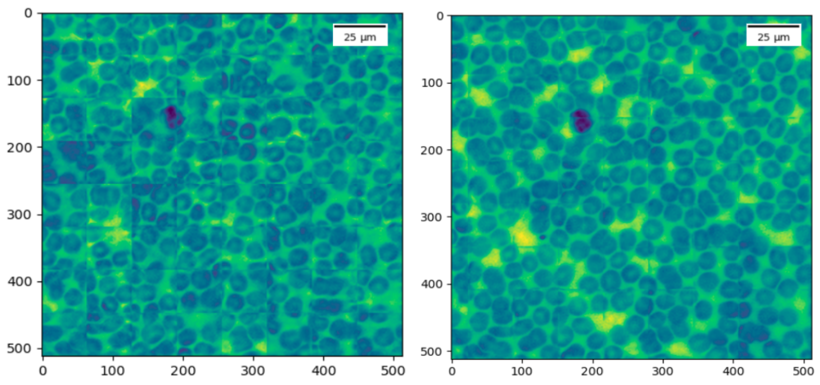


Fig. 13: Here the seam can be seen in the unpadded reconstruction (left) and is less visible in the padded reconstruction (right).

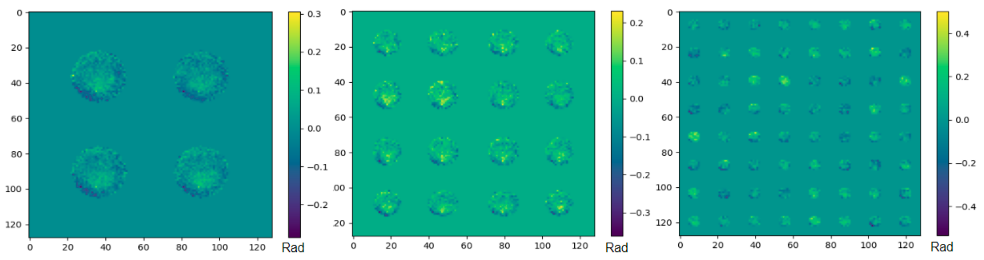


Fig. 14: Recovered pupils for different amounts of segmentation. More specific pupils are recovered as the image is segmented more.

3.2 DiffuserCam

Theory

Using the simple, inexpensive lensless optical system called DiffuserCam, it is possible to perform high resolution, 3D image reconstruction using a stack of point spread functions and a single-shot, 2D data measurement of the object. A simple linear combination problem is used to represent the problem, $\mathbf{b} = \mathbf{Hv}$, where $\mathbf{b}$ and $\mathbf{v}$ are vectors representing the 2D sensor measurement and the intensities of the 3D reconstructed image, respectively, and $\mathbf{H}$ is a forward model matrix whose columns consist of the caustic patterns (the PSFs of this system) at each point in the 3D field of view.

Obtaining each individual caustic pattern in the FOV would require an infinite number of measurements, therefore $\mathbf{H}$ is very large and difficult to define. $\mathbf{H}$ is also not necessarily invertible, so an ℓ_1 regularized, non-negativity-constrained, inverse optimization problem is required to solve for $\mathbf{v}$,

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v}\geq 0} \frac{1}{2} \parallel \mathbf{b} - \mathbf{Hv} \parallel_2^2 + \tau \parallel \Psi\mathbf{v} \parallel_1 \tag{1}$$

where τ is a tuning parameter that adjusts the degree of sparsity, Ψ is the identity matrix ($\Psi\mathbf{v}$ is mostly zeros), and $\parallel \Psi\mathbf{v} \parallel_1$ is the 3D total variation semi-norm, which reduces the noise in the image. In the system, an aperture is used so that the data measured on the sensor does not fill the entire sensor. Therefore, there is some space on the sensor on which theoretically no light is supposed to be picked up. The sparsity term helps us to get rid of the small amount of excess light that is picked up by the sensor.

The system is linear shift invariant, so compressed sensing is used to solve the underdetermined inverse problem. A convolutional forward model is used to compute $\mathbf{Hv}$. This avoids having to define $\mathbf{H}$ explicitly, which is extremely complex. Using this model also simplifies the calibration of the system because we assume shift invariance. In theory, only one calibration image of an on-axis point source is needed. However, in practice, capturing a stack of PSFs from different distances from the diffuser produces better results.

The inverse problem is then solved using iterative algorithms: Gradient Descent and the Alternating Direction Method of Multipliers (ADMM). However, since ADMM converges faster, it is the more efficient algorithm to use for 3D image reconstruction. Image reconstruction is performed in 300 iterations.

Hardware

The DiffuserCam setup consists of a diffuser placed at a set distance from a sensor, as seen in Fig. 15.

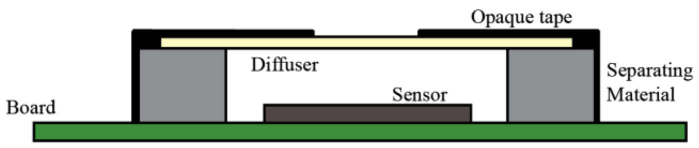


Fig. 15: The template setup for DiffuserCam. Image from DiffuserCam paper [3].

The diffuser can be any transparent, thin, smooth material, so we used scotch tape (as recommended in the camera building tutorial seen [10]). A Raspberry Pi Camera with the lens removed serves as our sensor. Note that it responds linearly to incident light, which is necessary for the system's linear shift invariance property. The diffuser refracts a point source of light, resulting in a caustic pattern PSF seen in Fig. 28. When an object is imaged, there is more than one point source of light being sent through, so it is equivalent to many different caustic patterns overlapping each other, which results in a blurred image of raw data, see Fig. 16.

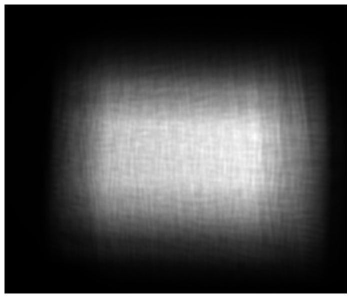


Fig. 16: The raw data measurement of a 3D object captured on the 2D sensor, from Antipa's 2018 paper [3].

The Raspberry Pi camera comes with a lens, so before capturing images we had to carefully remove the lens to expose the sensor. The sensor picks up the caustic patterns from calibration and the raw data measurement. In order to find where the PSF has the sharpest features, we must determine the distance of the ”focal” length of the diffuser. We did this manually by adding and taking away playing cards (any stackable objects can serve this function) to achieve the clearest caustic pattern. An aperture made from black cardboard is placed on the diffuser to limit the amount of light hitting the sensor. Our DiffuserCam can be seen in Fig. 17, attached to the far end of the cardboard box.

To calibrate the system, an LED placed in front of the cardboard pinhole aperture is used as a point source. The field of view of the system is defined through calibration, determining where the object being imaged should be placed. This is achieved by moving the point source laterally at different depths and observing the shift of the PSF, ensuring that the features of the object will be distinguished on the sensor. The LED point source is attached to the ruler that can be shifted easily along the cardboard box. A dark background is required for the best illumination of the object, so our images will be captured in a dark room inside the closed box. The PSFs and the raw data measurement are then downloaded to our code for 3D image reconstruction.

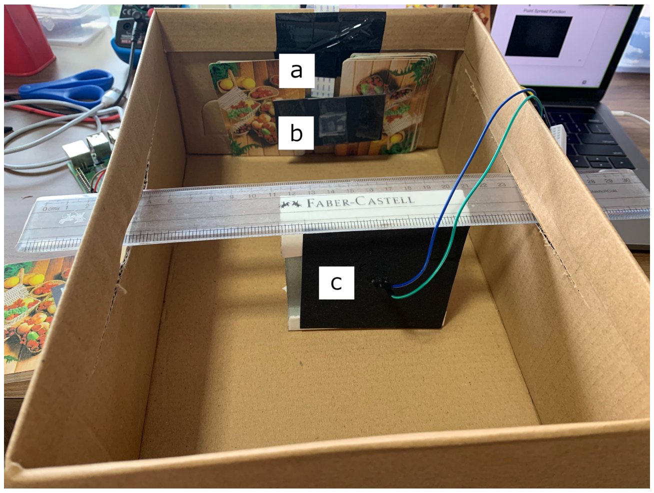


Fig. 17: Our setup for DiffuserCam. The RaspberryPi sensor (a) is placed at a set distance away from the diffuser, made from tape (b), which covers a small cardboard aperture. Playing cards are observed to the right and left of (a) and (b) to separate the two. Our LED point source (c) can be shifted closer and further away from the DiffuserCam using the ruler to take the PSF data measurements for system calibration.

Software

An online tutorial was created for users to build a DiffuserCam and implement the code for 2D image reconstruction. Both Gradient Descent and ADMM Colab notebooks are available online [11]. For 3D image reconstruction, the original DiffuserCam team created an open source code in MATLAB that executes ADMM in just 5 iterations, found on GitHub [12]. In order to make the code easier to use and understand, we implemented it in Colab, adding text explanations and extra comments within the code so future users can follow along easily. The code can be found here: `https://colab.research.google.com/drive/1aWyWku5FLhUjuIF8yXrUKQD32Vi1VSr1?usp=sharing`.

Using variable splitting, Lagrange multipliers, and penalty parameters, ADMM allows for a relatively fast 3D image reconstruction. It starts with the basic optimization problem stated in the Theory section:

$$\hat{\mathbf{v}} = \underset{\mathbf{v}\geq 0}{\arg\max} \frac{1}{2} \parallel \mathbf{b} - \mathbf{Hv} \parallel_2^2 + \tau \parallel \Psi \mathbf{v} \parallel_1 . \tag{2}$$

The minimization problem is then split into multiple minimization problems to better fit ADMM.

$$\begin{gathered}\hat{\mathbf{v}} = \underset{w\geq 0,u,x}{\arg\max} \frac{1}{2} \parallel \mathbf{b} - \mathbf{C}x \parallel_2^2 + \tau \parallel u \parallel_1 \\ \text{s.t. } x = \mathbf{Mv}, u = \Psi \mathbf{v}, w = \mathbf{v}.\end{gathered} \tag{3}$$

Here, the transfer function of DiffuserCam, **H**, has been split into two variables: **M**, the convolution, and **C**, the crop. The constraints imposed add more update steps to the algorithm.

Next, the *augmented Lagrangian* is formed, translating the equation to:

$$\begin{aligned} \mathcal{L}(u,x,w,\mathbf{v},\xi,\eta,\rho) = \frac{1}{2} \parallel \mathbf{b}-\mathbf{C}x \parallel_2^2 + \tau \parallel u \parallel_1 \\ + \frac{\mu_1}{2} \parallel \mathbf{M}\mathbf{v}-x \parallel_2^2 + \xi^T(\mathbf{M}\mathbf{v}-x) \\ + \frac{\mu_2}{2} \parallel \Psi\mathbf{v}-u \parallel_2^2 + \eta^T(\Psi\mathbf{v}-u) \\ + \frac{\mu_3}{2} \parallel \mathbf{v}-w \parallel_2^2 + \rho^T(\mathbf{v}-w) \\ + \mathbb{1}_+(w), \end{aligned} \tag{4}$$

where $\mathbb{1}_+(w)$ comes from the constraint $w \geq 0$:

$$\mathbb{1}_+(w) = \begin{cases} \infty, & w<0 \\ 0, & w \geq 0 \end{cases} \tag{5}$$

Finally, to minimize the objective function, the Lagrangian dual approach is to solve the optimization problem:

$$\underset{\xi,\eta,\rho}{\text{maximize}} \; \min_{u,x,w,\mathbf{v}} \mathcal{L}(\{u,x,w,\mathbf{v}\},\{\xi,\eta,\rho\}). \tag{6}$$

In this equation, we jointly minimize the primal variables $(u,x,w,\mathbf{v})$ first, and then maximize the dual variables (ξ,η,ρ). The algorithm then updates each individual variable. The updates are as follows:

$$\begin{aligned} u_{k+1} &\leftarrow \mathcal{T}_{\frac{\tau}{\mu_2}}(\Psi\mathbf{v}_k + \eta_k/\mu_2) \\ x_{k+1} &\leftarrow (\mathbf{C}^T\mathbf{C} = \mu_1 I)^{-1}(\xi_k \mathbf{M}\mathbf{v}_k + \mathbf{C}^T\mathbf{b}) \\ w_{k+1} &\leftarrow \max(\rho_k/\mu_3 + \mathbf{v}_k, 0) \\ \mathbf{v}_{k+1} &\leftarrow (\mu_1 \mathbf{M}^T\mathbf{M} + \mu_2 \Psi^T\Psi + \mu_3 I)^{-1} r_k \\ \xi_{k+1} &\leftarrow \xi_k + \mu_1(\mathbf{M}\mathbf{v}_{k+1} - x_{k+1}) \\ \eta_{k+1} &\leftarrow \eta_k + \mu_2(\Psi\mathbf{v}_{k+1} - u_{k+1}) \\ \rho_{k+1} &\leftarrow \rho_k + \mu_2(\mathbf{v}_{k+1} - w_{k+1}) \\ &\text{where} \\ r_k = (\mu_3 w_{k+1} - \rho_k &+ \Psi^T(\mu_2 u_{k+1} - \eta_k) + \mathbf{M}^T(\mu_1 x_{k+1} - \xi_k) \end{aligned} \tag{7}$$

For more background information on ADMM, refer to [13], sections 2 and 3 of [14], and Chapters 5 and 9 of [15].

3.3 Speckle Imaging

Theory

A speckle pattern is formed when a coherent light source like a laser passes through a scattering media. The light will be randomized by the scattering media, and as the light propagates through space, the random components will interfere with each other, resulting in a random intensity fluctuation known as a speckle [16]. A typical speckle pattern is as shown in Fig. 18. The scattering of light waves makes it impossible to see through scattering medium. However, the memory effect states that when light is transmitted through a medium, the phase of the optical field is remembered during transmission. This means that we can still extract information from speckle patterns. However, the memory effect has a very strict constrain on the objects:

$$\Delta x \ll u \cdot \lambda / \pi L$$

where u is the distance between object and the scattering medium, Δx is the distance between the end points on the object, λ is the laser wavelength, and L is thickness of the diffuser.

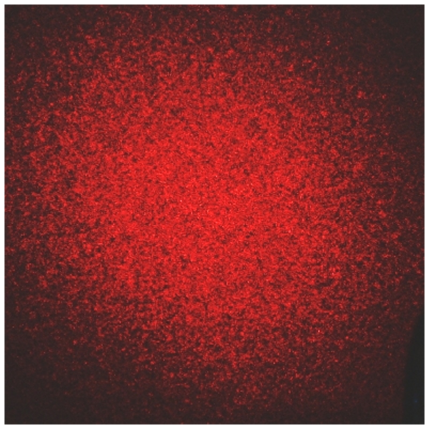


Fig. 18: Example of a typical speckle pattern captured by a camera [17].

Although speckle patterns seem to have a disorderly nature, it does have some important statistical properties. Distribution of the intensity of a speckle pattern should be negative exponential, phase distribution should be uniform, and the amplitude should be Rayleigh distributed [16]. We can use these properties to make sure that acquired speckle patterns are reasonable by fitting the magnitudes of the pixels to a negative exponential curve.

In our speckle imaging system, we illuminate an object with the speckle pattern produced by a laser and a diffuser, our chosen scattering medium due to its accessibility. Then a sensor will capture, in mathematical terms, the convolution between the speckle pattern function and the object function. This results in another random, speckle-like image. In order to recover the original image of the object, we first take the autocorrelation of the captured image. This process is shown in Fig. 19.

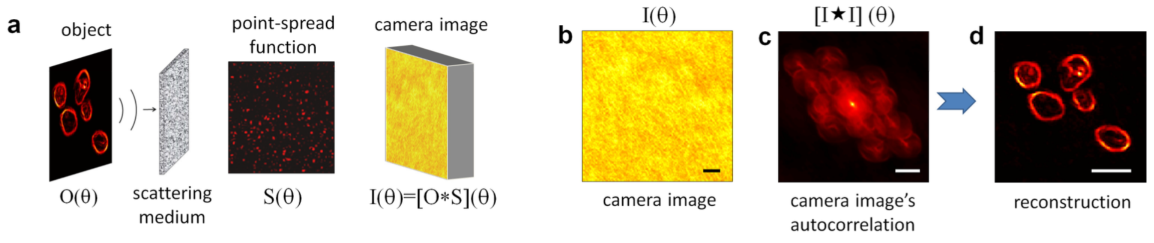


Fig. 19: Speckle imaging system concept. a) An object is hidden behind a scattering medium that produces a speckle pattern as its PSF. b) A camera captures the convolution between the hidden object and speckle pattern. c) Autocorrelation and phase retrieval algorithms performed to reconstruct object image. Figure from Katz's 2014 paper [4].

The reason why we take the autocorrelation of the captured image is to get rid of the presence of the speckle pattern. Autocorrelation of the image can be expressed as:

$$I \bigstar I = [(O * S) \bigstar (O * S)] = [(O \bigstar O) * (S \bigstar S)] = \delta * (O \bigstar O) = O \bigstar O \tag{8}$$

In Eq. 8, the captured image I can be rewritten as the convolution between the object O and the speckle function S; autocorrelation is denoted as $\bigstar$. According to the convolution theorem, we can rewrite the autocorrelation of $(O * S)$ as the autocorrelation of O convolved with the autocorrelation of S [4]. Since the autocorrelation of a speckle function is a delta function shown in Fig. 20, the result is simply the object's autocorrelation, shown in Fig. 21. This is because the convolution of a delta function with any function results in said function. In fact, mathematically, the convolution between a delta function and object function is how lenses operate because lenses can be represented as a delta function. This creates the point to point mapping property that lenses have. For a speckle imaging system however, after autocorrelation, we still need to recover the object with phase retrieval methods.

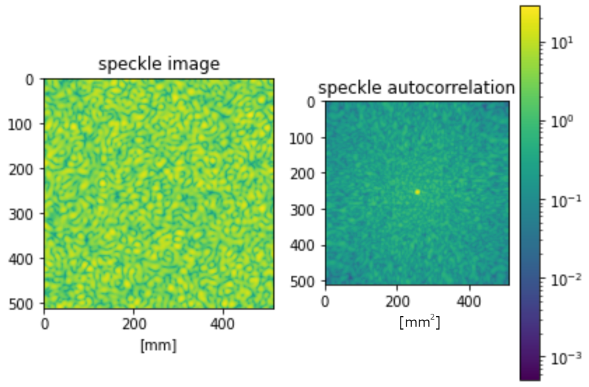


Fig. 20: The autocorrelation of a speckle pattern is a delta function.

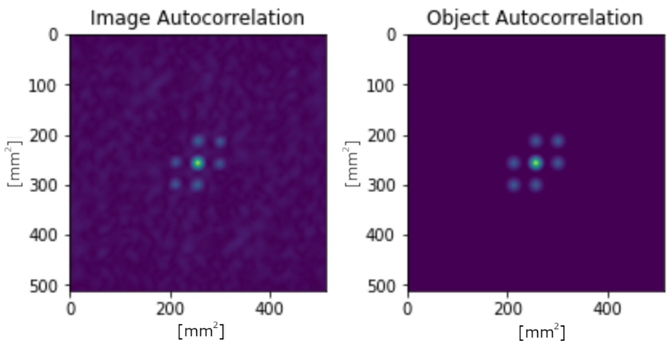


Fig. 21: The captured image's autocorrelation matches the object's autocorrelation with a minimal background noise. This shows that autocorrelation can eliminate most speckle pattern presence.

Hardware

For our speckle imaging system, we used a 5mW 650nm laser pointer as our coherent light source. We used a laser different from the original paper's [4] 532nm laser for safety purpose as the experiment was performed at home. We mounted our devices onto an optical bench from an Eisco optical set to align the optical instruments. We used two biconvex lenses to form a Galilean telescope in order to magnify the speckle sizes to be greater than the camera pixel size. In terms of capturing images, we placed a Raspberry Pi camera off the axis, in order to get away from the ballistic light in the middle, which will result in saturated pixels. We used a short camera exposure time and a low ISO number in order to capture all the speckle patterns.

In order to reduce the background noise, we set up our devices in a room without windows. Raspberry Pi Zero's activity light will be on while usage. Thus, we covered the act light on the Raspberry Pi Zero with black tape. We covered the white frames in Eisco optical set for a similar reason.

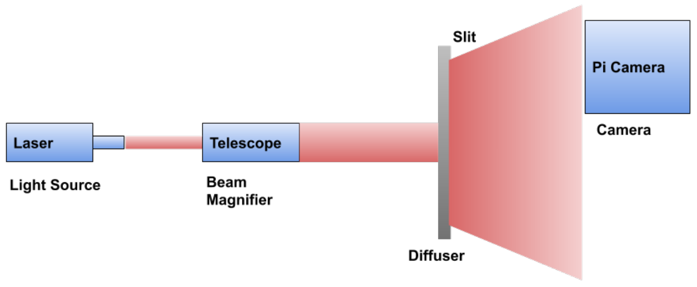


Fig. 22: Initial physical setup of the speckle imaging project.

Fig. 22 shows our initial experimental setup. We used a telescope to expand the input laser beam in order to decrease speckle size and improve image resolution. The diffuser in the setup would scatter light waves, resulting in a speckle pattern. The speckle pattern then illuminated the object and propagated the image to the camera. The small speckle size later emerged as a problem, as the camera cannot capture a speckle grain smaller than its pixel. In order to increase the speckle size, we changed our physical setup to the one shown in Fig. 23. We stopped using the telescope to stop producing an expanded input beam. We also added a slit after the diffuser and added the magnifying telescope between the slit and camera, all in pursuit of increasing speckle size.

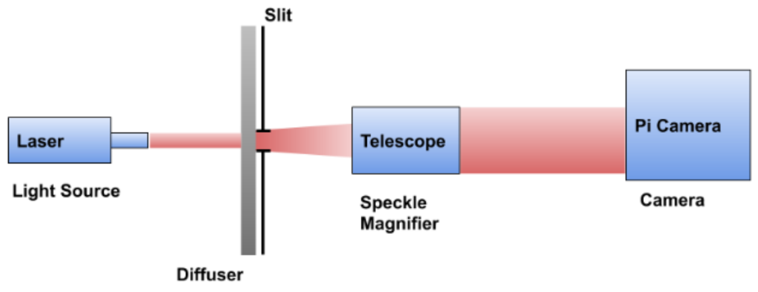


Fig. 23: Physical setup aimed at having bigger speckle patterns.

After making sure we were getting a reasonable speckle pattern with this physical setup, we placed an object to image between two diffusers, all located before the slit. See Fig. 24. The distance between diffuser and camera is 5cm so we can capture as much information as possible, and the distance between the object and the diffuser is 50cm in order to enlarge the diameter of memory effect.

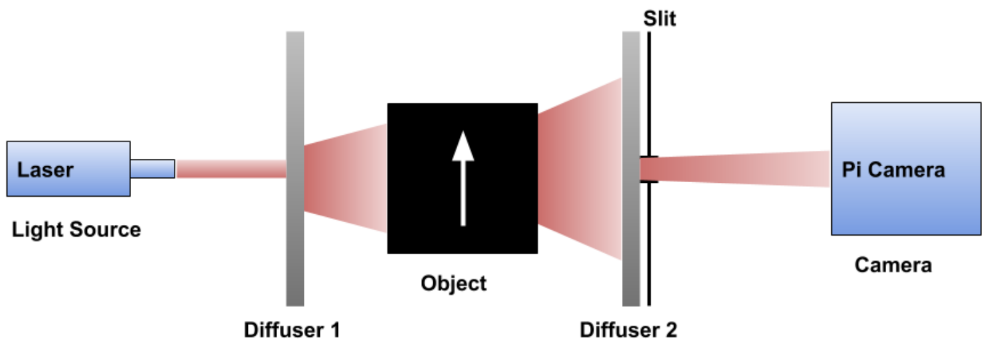


Fig. 24: Final physical setup to image objects.

Software

In parallel to building the speckle imaging setup, we developed simulations of the setup in Google Colab to gain a baseline of what results to expect. This involved an exploration into light propagation methods, such as Angular Spectrum (AS) and numerical calculation of the Rayleigh-Sommerfeld (RS) diffraction integral.

Angular spectrum is commonly used for near-field propagation simulation. The AS method treats the propagation of light as a superposition of plane waves with different wave vectors and uses a Fourier transform to compute the light fields in the spatial-frequency domain [18]. In the AS method, we define $U(x_m, y_n, 0)$ as the initial light field, and $U(x, y, z)$ as the output light field. We also define $G(\alpha, \beta, z)$ as the ”optical transfer function”. An optical transfer function defines how lights of different frequencies will react to the propagation process. The idea of the AS method is to disassemble the input light field into plane waves, convolve each of them with the optical transfer function, and add them together to get the propagated light field. We know convolution can be performed by FFT and IFFT, which are linear operations. Thus, we do not have to actually perform the disassemble and addition, and convolving the whole input light field with the optical transfer function will give the same result. The algorithm is then given as

$$U(x,y,z) = IFFT2\{FFT2\{U(x_m,y_n,0)\}\cdot\times G(\alpha_m,\beta_n,z)\}$$

where $G(\alpha,\beta,z) = exp(j\sqrt{k^2-\alpha^2-\beta^2}z)$. Here, α and β are coordinates in the Fourier domain and k is the wave number [18].

The direct integration (DI) method is usually used for far-field propagation. The DI method computes the diffraction integrals in the spatial domain by means of numerical integration, which can be treated as a linear convolution and therefore can be calculated with FFT and IFFT [18]. We have the Rayleigh-Sommerfeld diffraction integral formula as:

$$U(x,y,z) = \iint_A U(\varsigma,\eta,0)g(x-\varsigma,y-\eta,z)d\varsigma d\eta = \iint_A U(\varsigma,\eta,0)\frac{exp(jkr)}{2\pi r}\frac{z}{r}\times(\frac{1}{r}-jk)d\varsigma d\eta$$

where $r = \sqrt{(x-\varsigma)^2+(y-\eta)^2=z^2}$. In order to calculate the integral numerically, we sample the $U(\varsigma,\eta,0)$ to $N\times N$ equidistant grids. We then use the Riemann sum to calculate the integral:

$$U(x_m,y_n,z) = \sum_{i=1}^{N}\sum_{i=1}^{N}U(\varsigma_i,\eta_j,0)g\times(x_m-\varsigma_i,y_n-\eta_j,z)\Delta\varsigma\Delta\eta$$

where $\Delta\varsigma$ and $\Delta\eta$ are sampling intervals on the observation plane. The Riemann sum in the equation above can be seen as a convolution between the initial light field $U(\varsigma_i,\eta_j,0)$ and the optical transfer function $g(x_m,y_n,z)$, similarly to the AS method. Therefore, we can calculate the output light field as

$$U(x,y,z) = IFFT\{FFT\{U(x,y,0\}\}\cdot\times FFT\{H\}]\Delta\varsigma\Delta\eta,$$

U and H here are defined in [18] at equations (11) and (12).

During the image reconstruction process, the resolution of the reconstruction is directly related to the size of the speckle grains. Controlling speckle size is beneficial, so we developed programs that also allow us to engineer various speckle patterns and sizes. We witnessed that larger input beams produced smaller speckle grain sizes and vice versa. Furthermore, this program also allows us to analyze the statistics of speckle mentioned in the Theory section. This will help us decipher speckle patterns from noise in the actual physical experiment.

After capturing data with the Raspberry Pi camera, the information will be processed with reverse autocorrelation based phase retrieval algorithms. As mentioned in Theory, we perform autocorrelation on the captured image to get rid of speckle patterns and to obtain the original object's autocorrelation. After, we need to recover the object from its autocorrelation. We followed the same image processing methods as in the paper [4], which followed Bertolotti et al.'s [19] speckle experiment. First, we applied a Tukey windowing function $W(x,y)$ to our autocorrelation image $I(x,y)$. Then, we Fourier transformed the result to obtain the object's power spectrum S_{meas}:

$$S_{meas}(k_x, k_y) = |FT\{W(x,y)I(x,y)\}|$$

We ran the Hybrid-Input Output algorithm on the power spectrum function in order to quickly fit constraints within the algorithm, followed by several iterations of the Error Reduction method to ensure that the result will converge. Specifically, the HIO algorithm is ran with a decreasing beta factor from β = 2 to 0, in steps of 0.04. For each β value, 40 iterations of the algorithm is performed, with the resulting image processed with ER for an additional 40 iterations. The object constraints used were real and non-negativity, which is mostly the case in microscopy.

Location of all simulations and code can be found `https://bit.ly/39K8G79`.

4 Results

4.1 UC2 FPM

Initial trials with the system offered very poor reconstructions from the transparency images. We noticed that the sample factor for the system was very low which was causing a very blurry reconstruction. Our sample factor was calculated at 0.38 whereas [20] suggests a sample factor of at least 2. After discovering this, we readjusted the spacing of the modules to include more spacing in three places. The distance between the LED array and the z-stage was increased to 132mm, the distance between the sample and the MO was set at 78mm, and the distance between the MO and the HQ Raspberry Pi Camera was changed to 134mm. This increased our sample factor from 0.38 to 2.29.

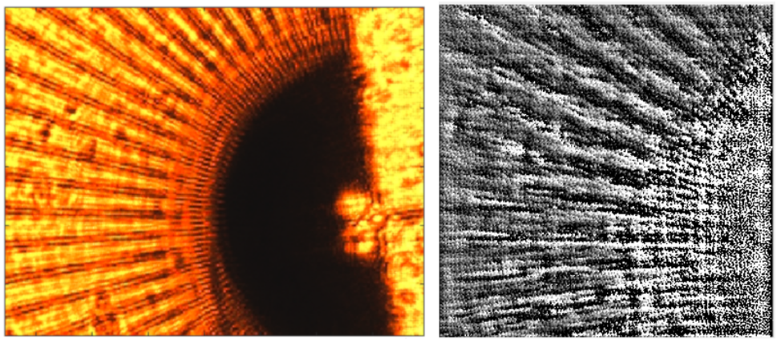


Fig. 25: Subregion of an acquired brightfield image of a transparency (left) and the low-sample-factor reconstruction (right).

Many of the acquired images from LEDs at a high angle with the sample are very dark. With low contrast, this caused us to lose the information encoded in some of these ”darkfield” images. To increase this contrast and preserve the information in these images, we opted to use a High Dynamic Range (HDR) to capture the images. Without doing this, only a few LEDs from the center of the matrix would offer enough illumination to be useful for image reconstruction.

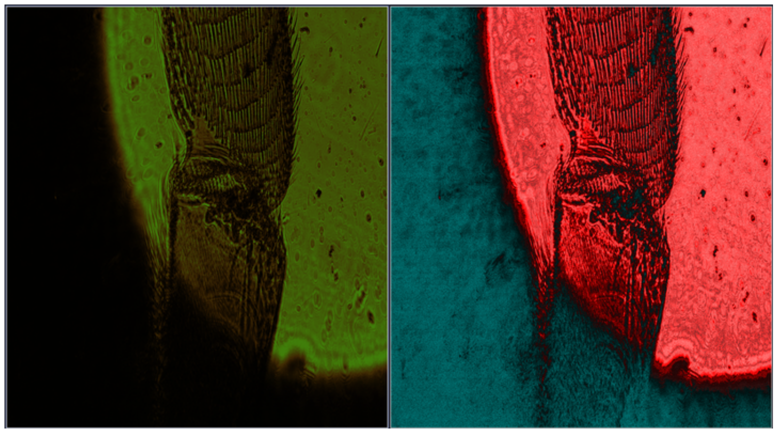


Fig. 26: Acquired brightfield images of a honeybee leg. Left) SDR and Right) HDR.

From these HDR intensity images, we were able to recover both intensity and phase images of a honeybee leg. Comparing these reconstructions to the low-resolution acquired image, we can see how much detail was recovered as the resolution was increased by a factor of four. The phase image is also useful for noticing depth differences in the sample. For example, we can see that the center of the leg is relatively flat and there is an area which bulges out towards the top. This type of topographic analysis would not be available without the phase retrieval built into the FPM algorithm.

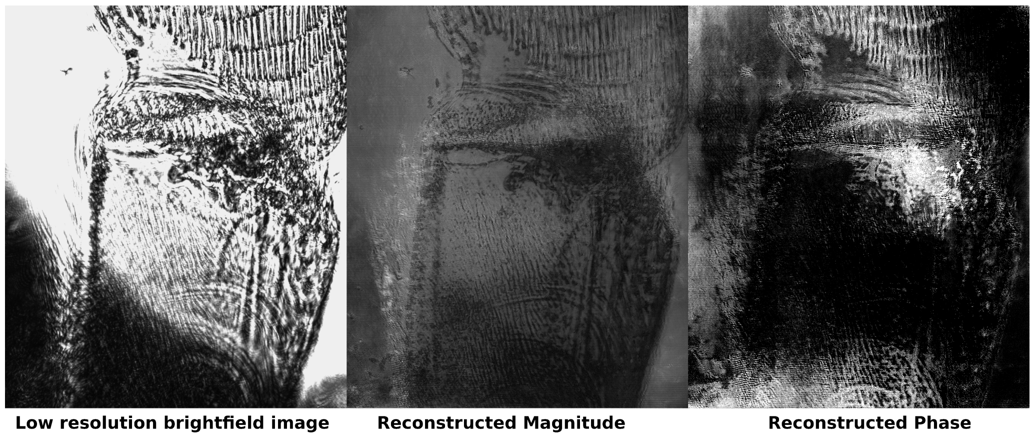


Fig. 27: Left) Intensity of acquired image (2000x2000 px), Middle) Reconstructed image from FPM (8000x8000 px), and Right) Recovered phase information of image (8000x8000 px)

Overall, the UC2 FPM system has proven to work well. We have demonstrated the accessibility of the system so that future research in this sub-field of computational imaging is possible.

4.2 Diffuser Cam

We have finished setting up our system and are currently working on calibration. We are not yet seeing caustic patterns, which we believe is due to our LED pinhole aperture letting too much light through, which exposes the sensor to too much light. An example of our calibration images obtained so far can be seen in Fig. 28. More work needs to be done on our setup to capture images.

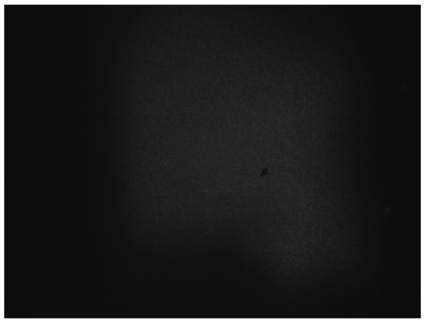


Fig. 28: This measurement is a result of an LED set at low brightness placed in front of a pinhole aperture a set distance away from the DiffuserCam. We need to continue adjusting the aperture and brightness of the LED in order to get the caustic pattern we are looking for.

As shown in Fig. 29 (next page), we were able to reconstruct the USAF target image with the open source example data obtained from [3]. Our results are at a lower resolution compared to that of the Matlab code version (see Fig. 30), suggesting that the Python code we have created still needs some debugging. There is more work to be done with increasing the speed and efficiency of our code.

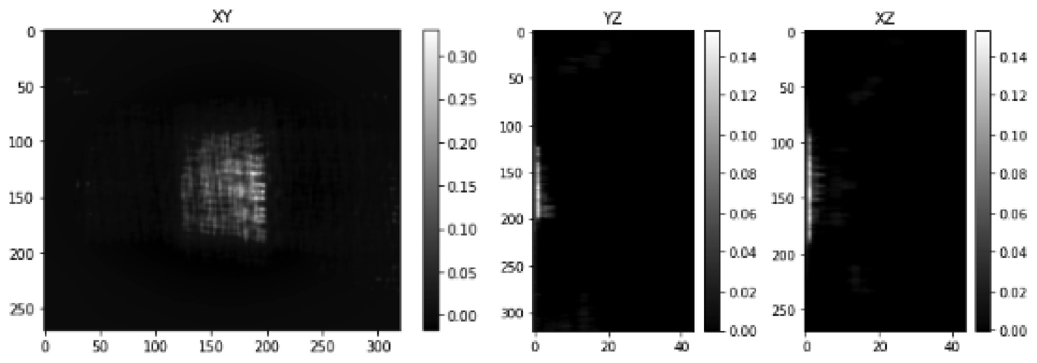


Fig. 29: 3D image reconstruction after 300 iterations in Python. Units in px.

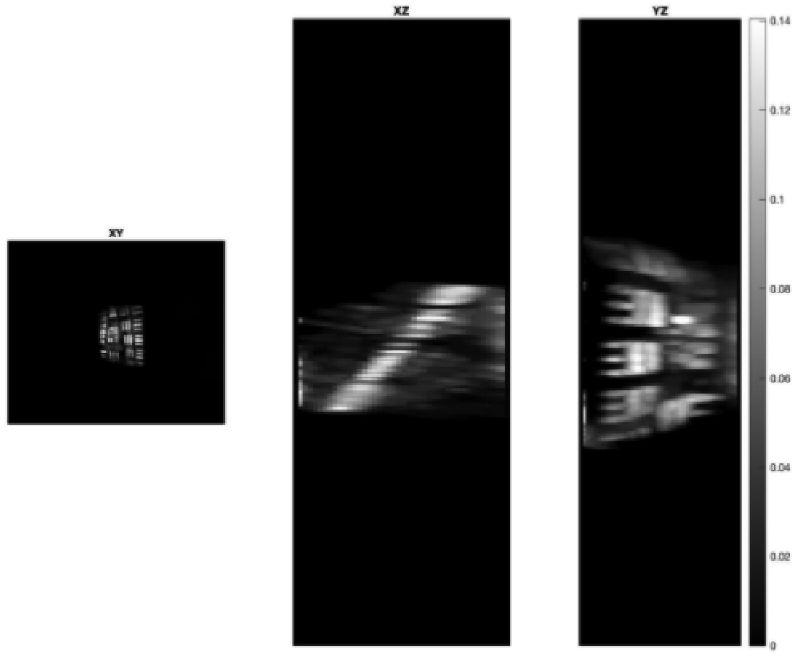


Fig. 30: 3D image reconstruction after 300 iterations in Matlab. Units in px.

4.3 Speckle

Simulation

As shown in Fig. 31, we have successfully simulated the experimental setup with a valid result as the object reconstruction looks identical to the original object. This provides proof of concept for the speckle imaging system. In the simulation, we used a matrix with random values to mimic an actual speckle pattern. Using a random matrix also optimizes the resolution of our image, as the speckle size is the same as the pixel size. We used various propagation methods when necessary. The captured image was simulated as the convolution between the object function and the random matrix. Once we took the captured image's autocorrelation using FFT and IFFT methods, we were able to reconstruct the original object with phase retrieval algorithms with the goal to reverse autocorrelation.

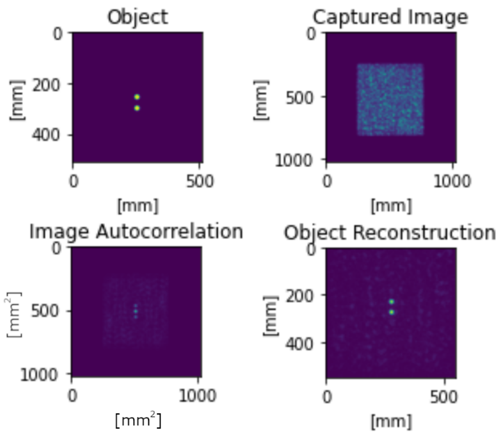


Fig. 31: Simulation Result with a two circle object. Captured image is convolution between object and speckle pattern. Original object information can be observed with the image autocorrelation. The object reconstruction closely matches to original object.

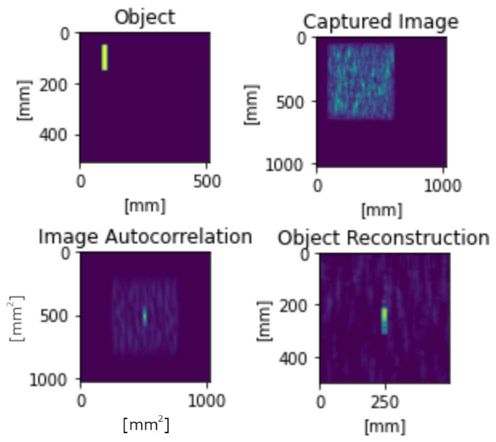


Fig. 32: Simulation Result with a rectangle object. The object reconstruction was padded to increase the image quality, and the object was shifted manually to the center. There is some gradient color pattern occurring within the rectangle, suggesting a less complete reconstruction. This speaks to the object dependent characteristic of computational imaging systems.

We were able to successfully reverse autocorrelation using methods mentioned in the Methods section. However, there is some discussion to be had around the efficacy of the algorithms we used. While the object reconstruction in Fig. 31 looks nearly identical to the original object, this isn't necessarily the case for all objects. For computational imaging systems, it is often the case that quality of reconstructed images are object dependent. For lens imaging system, the quality of images is system dependent. This means that reconstruction for objects with little sparsity or many details will result in a poorer reconstruction. The two circle object we chose to simulate in Fig. 31 is quite simple and has enough sparsity to produce a successful reconstruction. However, the rectangle object in Fig. 32 results in a slightly poorer reconstruction because of its lessened sparsity.

Furthermore, in each phase retrieval algorithm, we only used spatial domain constraints of realness and positivity, but perhaps adding appropriate Fourier domain constraints might further improve reconstruction quality.

It is also important to consider the time that the reconstruction takes. Analyzing the efficiency of our reconstruction methods was the not the focus of our project, but it is important to note that we observed algorithm running times of several minutes, sometimes hours depending on the complexity of image. We also observed that running algorithms using local servers such as Jupyter notebook, rather than shared servers such as Google Colab, sometimes increased the running speed.

Experiment

For the physical experimental setup, we were able to build the speckle imaging system. We have been able to capture a speckle pattern by shining the laser through a diffuser, as shown in Fig. 33. We have reduced background noise to a reasonable level, and we have been able to image objects shown in Fig. 34 and Fig. 35. However, we are still in the process of capturing object images with higher complexity, since the size of object is constrained by the memory effect diameter:

$$\Delta x \ll u \cdot \lambda / \pi L$$

where $u = 50cm$ is the distance between the object and the diffuser, Δx is the distance between points on the object, $\lambda = 650nm$ is the laser wavelength, and $L = 1mm$ is thickness of the diffuser. We therefore need an object with size smaller than 0.01cm; in the original paper [4], the author used an object with 40um diameter.

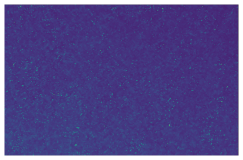


Fig. 33: Speckle pattern captured with our physical setup.

We first imaged an object of a circle with a 1 mm diameter because we didn't have access to smaller objects in a non-laboratory setting. As shown in Fig. 34, the circle shape can be resolved in the object reconstruction. The image during the processing stage has also been padded to produce a clearer reconstruction. However, because the circle shape is simplistic, we performed another trial

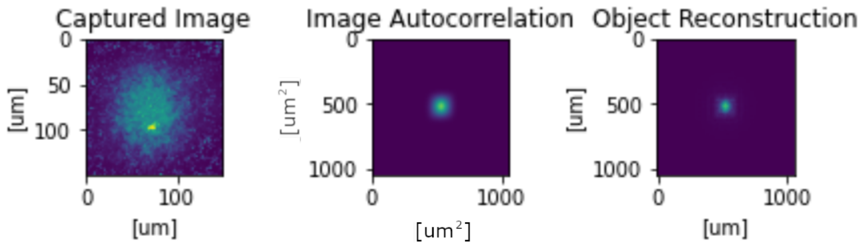


Fig. 34: Experimental result of a circle object. The circle can be clearly seen, which indicates a successful reconstruction. However, due to the simplistic nature of the object, more trials should be run to make sure that the reconstruction is not an artefact.

on a different, slightly more complex object of two dots to make sure that the circle reconstruction is not just an artefact.

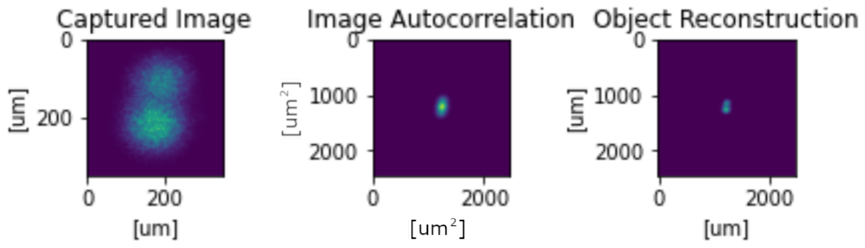


Fig. 35: Experimental result of an object of two dots. Two dots can be made out but the edges of the dots are not distinct.

As shown in Fig. 35, the object reconstruction was successful as you can decipher out two circles, which gives us confidence that the single circle reconstruction was also successful. The image quality of the reconstruction could definitely be improved upon as there aren't distinct edges of the two circles. The blurry quality of both reconstructions could also be attributed to the fact at both object diameters of 1 mm were outside the memory effect limitation. This means that the objects we imaged were too large in size to produce completely accurate reconstructions.

Overall, our results show that our speckle imaging system and the reconstruction algorithms work but there are many limitations, such as image processing times and object diameter limits. More work should should be done to have the imaging system perform at its full potential.

5 Potential Future Work

5.1 UC2 FPM

The FPM algorithm lends itself to many different imaging modalities. Future work could include implementing some of these with our UC2 system such as Fourier Ptychographic Tomography for thick sample imaging and the recovery of complex indices of refraction [21].

Work can also be done to improve image acquisition time as currently, exposing each of the images and uploading them can take up to an hour. Future students might consider implementing Multiplexed FPM [22] and Self-Learning FPM [23] to improve acquisition time and reduce unnecessary redundancy in acquired data.

5.2 Diffuser Cam

There has been work done in the incorporation of Deep Learning with ADMM [24] to increase computation speed and image quality. Moving forward, we would expect to explore the use of Machine Learning techniques with gradient descent. We have been reading papers such as [25] and [26] and plan to conduct an in depth literature review to gain a better understanding on previous work in the field.

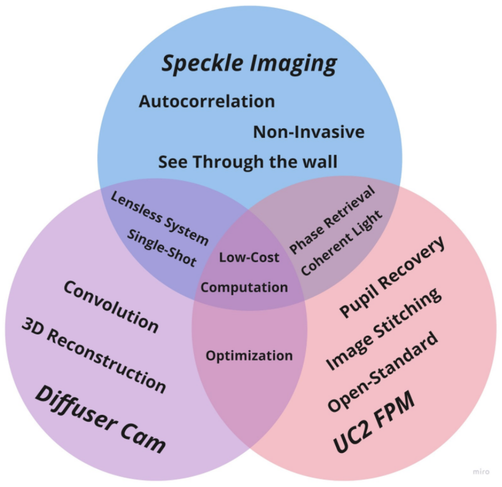
5.3 Speckle

For future work in a lab, imaging objects within the memory effect diameter limit is definitely a top priority. Being able to do this will help produce better reconstructions and continue to showcase proof of concept of our system. Furthermore, although not necessary, it might be helpful to also use a laser driver with TTL control in order to control the laser brightness.

As mentioned previously, the HIO and ER phase retrieval algorithms have some limitations to creating a high quality reconstruction. More in-depth analysis in terms of the object sparsity constraints, Fourier domain constraints, and running time will be beneficial to determining the efficacy of these algorithms. In the original paper [4], the authors mentioned that they aren't able to overcome some common stagnation and artefacts due to the simplicity of the algorithms for the physical experimental results. Another reconstruction method to deploy is deep learning neural networks for more efficient image processing. Other advanced phase retrieval algorithms to explore include triple-correlation analysis, Knox-Thompson, and exponential filtering.

For continual development of the speckle imaging system, there shows promise in combining the penetration power of an ultrasound scan and the precision of an optical microscopy. Blending all these imaging techniques together could produce a computational imaging system that results in even higher resolution and higher imaging penetration depth.

6 Conclusion



Our summer research serves as proof of concepts for the computational imaging systems: UC2 FPM, DiffuserCam, and Speckle Imaging. These imaging systems have cleverly overcome resolution, accessibility, and utility limitations of the conventional lens imaging system by utilizing the abilities of a computer. The final costs of each project was under $300 (not including the cost of a 3D printer for the UC2 FPM project), making each systems accessible to wide range of interested individuals. Because we performed each experiment at home and had to purchase all of the necessary optical instruments, this price point can decrease even further as most of the instruments used can be found in a lab. This makes research and development into computational imaging and tools for microscopy accessible to students as well. Computational imaging does have its limitations. There is still much work to be done in the field but our work this summer has proven that computational imaging has enormous potential in transforming the way we take images.

Acknowledgements

We would like to thank the National Science Foundation (NSF) for providing funding for this summer experience, and Harvey Mudd College for hosting us (remotely). Thank you to Professor Josh Brake at HMC for advising all three projects and being an incredible mentor, and to Professor Susan Martonosi and Professor Lisette De Pillis at HMC for organizing this wonderful REU experience. Special thanks to the rest of our team members Alec Vercruysse, Rosey Sams, and Shreya Sanghai at HMC who contributed a significant amount to these research projects and helped co-author and edit several sections of this report. We would also like to acknowledge Maya Watanabe, Christina Catlett, Daniel Shenker, and Daniel Ralston for helping edit an early draft of our report.

References

[1] Zheng, G., Horstmeyer, R., & Yang, C. ”Wide-field, high-resolution Fourier ptychographic microscopy.” *Nature Photonics.* (2013).

[2] Diederich, B., Lachmann, R., Carlstedt, S., Marsikova, B., Wang, H., Uwurukundo, X., Mosig, A., & Heintzmann, R. ”UC2 – A Versatile and Customizable low-cost 3D-printed Optical Open-Standard for microscopic imaging.” (2020).

[3] Antipa, N., Kuo, G., Heckel, R., Mildenhall, B., Bostan, E., Ng, R., & Waller, L. ”DiffuserCam: lensless single-exposure 3D imaging.” *Optica.* (2018).

[4] Katz, O., Heidmann, P., Fink, M., & Gigan, S. ”Non-invasive real-time imaging through scattering layers and around corners via speckle correlations.” *Nature Photonics.* (2014).

[5] Fienup, J.R. ”Phase retrieval algorithms: a comparison.” *Appl. Opt.* 21, 2758-2769 (1982).

[6] Shechtman, Y., Eldar, Y.C., Cohen, O., Chapman, H.N., Miao, J., & Segev, M. ”Phase Retrieval with Application to Optical Imaging.” (2014).

[7] STICKY: Is your Pi not booting? (The boot problems sticky) - Raspberry Pi forums. Retrieved July 20, 2020, from `https://bit.ly/3jqdjI7`

[8] Diederich, B., Lachmann, R., Carlstedt, S., Marsikova, B., Wang, H., Uwurukundu, X., Mosig, A., & Heintzmann, R. ”UC2 - A Versatile and Customizable low-cost 3D-printed Optical Open-Standard for microscopic imaging.”(2020). 10.1101/2020.03.02.973073.

[9] Ou, X., Zheng, G., & Yang, C. ”Embedded pupil function recovery for Fourier ptychographic microscopy.” *Opt. Express.* 22, 4960-4972 (2014).

[10] Biscarrat, C., et al. ”How to Build a (Pi) DiffuserCam.” Waller Lab, 12 Dec. 2018, `https://waller-lab.github.io/DiffuserCam/tutorial/building_guide.pdf`.

[11] Antipa, Nick, et al. “Build Your Own DiffuserCam: Tutorial.” DiffuserCam, Waller Lab, 2018, `waller-lab.github.io/DiffuserCam/tutorial.html`.

[12] Waller-Lab. “Waller-Lab/DiffuserCam.” GitHub, 2017, `github.com/Waller-Lab/DiffuserCam`

[13] Biscarrat, Camille, et al. Diffuser Cam: Lensless Imaging Algorithms. Waller Lab, 10 Dec. 2018, `waller-lab.github.io/DiffuserCam/tutorial/algorithm_guide.pdf`

[14] Boyd, S., et al. ”Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.” *Founders and Trends in Machine Learning.* (2011). `http://stanford.edu/~boyd/papers/pdf/admm_distr_stats.pdf`

[15] Boyd, S. P., & Vandenberghem, L. ”Convex Optimization.” *Cambridge University Press.* (2018). `https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`

[16] Goodman, J. W. ”Some fundamental properties of speckle.” *J. Opt. Soc. Am.* 66, 1145-1150 (1976).

[17] Speckle pattern. (2002, February 26). *Wikipedia, the free encyclopedia.* Retrieved July 29, 2020, from `https://shorturl.at/guwCN`

[18] Shen, F., & Wang, A. ”Fast-Fourier-transform based numerical integration method for the Rayleigh-Sommerfeld diffraction formula.” *Appl. Opt.* 45, 1102-1110 (2006).

[19] Bertolotti, J., van Putten, E., Blum, C. et al. ”Non-invasive imaging through opaque scattering layers.” *Nature.* 491, 232–234 (2012). `https://doi.org/10.1038/nature11578`

[20] Aidukas, T., Eckert, R., Harvey, A.R. et al. Low-cost, sub-micron resolution, wide-field computational microscopy using opensource hardware. Sci Rep 9, 7457 (2019). `https://doi.org/10.1038/s41598-019-43845-9`

[21] Horstmeyer, R., Chung, J., Ou, X., Zheng, G., & Yang, C. ”Diffraction tomography with Fourier ptychography.” *Optica* 3, 827-835 (2016).

[22] Tian, L., Li, X., Ramchandran, K., & Waller, L. ”Multiplexed coded illumination for Fourier Ptychography with an LED array microscope.” *Biomed. Opt. Express.* 5, 2376-2389 (2014).

[23] Zhang, Y., Jiang, W., Tian, L., Waller, L., & Dai, Q. ”Self-learning based Fourier ptychographic microscopy.” *Opt. Express.* 23, 18471-18486 (2015).

[24] Monakhova, K., Yurtsever, J., Kuo, G., Antipa, N., Yanny, K., & Waller, L. ”Learned reconstructions for practical mask-based lensless imaging.” *Opt. Express.* 27, 28075-28090 (2019).

[25] Yang, F., Pham, T., Gupta, H., Unser, M., & Ma, J. ”Deep-learning projector for optical diffraction tomography.” *Opt. Express.* 28, 3905-3921 (2020).

[26] Barbastathis, G., Ozcan, A., & Situ, G. ”On the use of deep learning for computational imaging.” *Optica.* 6, 921-943 (2019).