

Compressive Imaging Systems

CS 763, Lectures Notes


PART 1: RICE SINGLE PIXEL CAMERA

Standard Camera

- Consists of an aperture, a lens and a detector array.
- Light from the scene enters camera through aperture and is focussed onto detector array by the lens.
- Number of pixels on detector array = size of the image.

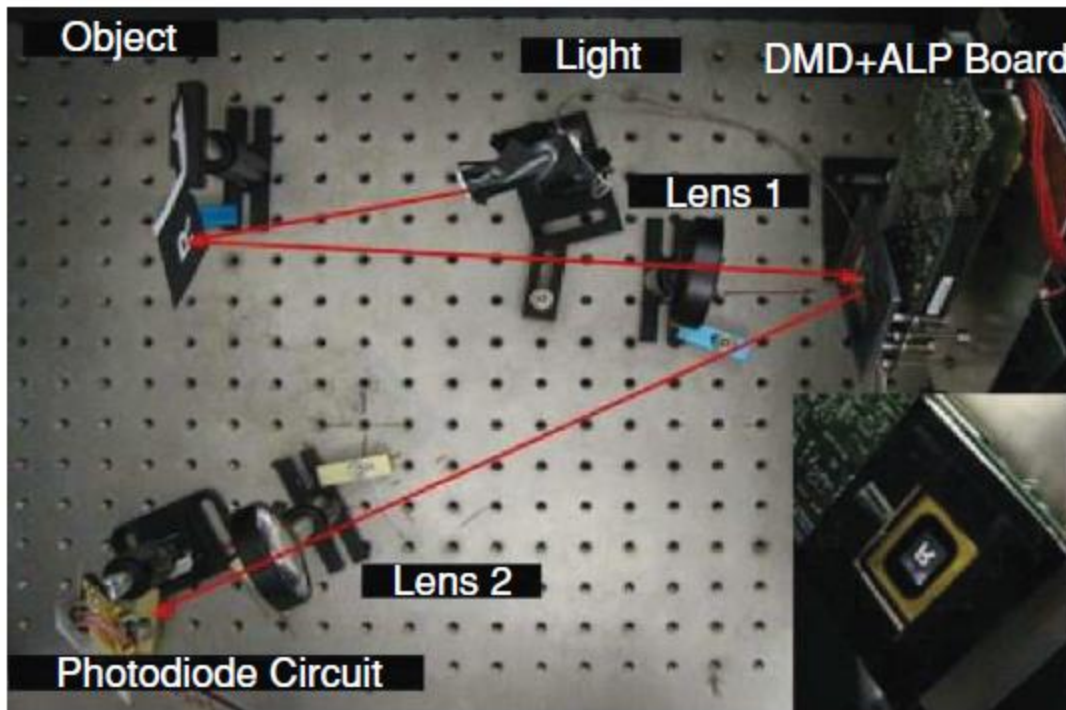
Rice Single Pixel Camera: a CS-based camera

- It does not measure the image 2D array.
- Instead it directly measures the dot-product of the image with a set of random codes. This is mathematically shown below:


$$\forall i, 1 \leq i \leq m, y_i = \mathbf{f}^T \boldsymbol{\phi}_i$$

- We now want to reconstruct \mathbf{f} given the set of measurements, i.e. $\{y_i\}$ and random codes.

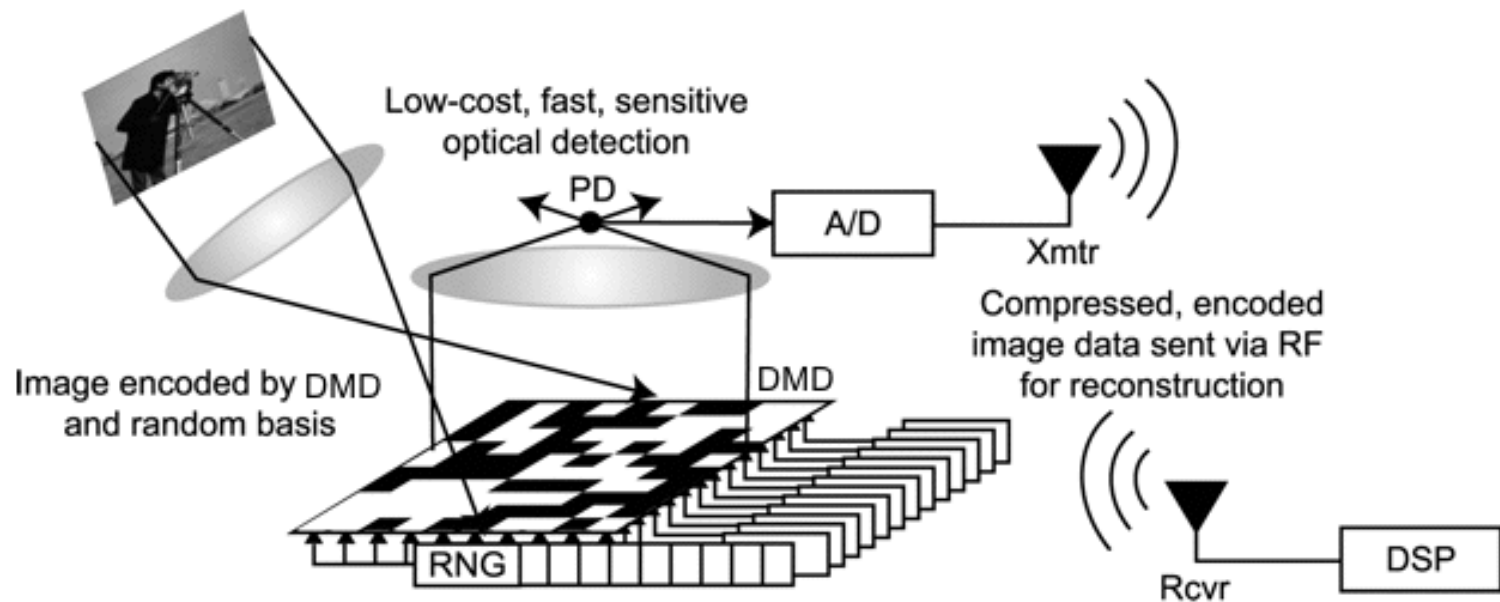
Rice Single Pixel Camera: a CS-based camera



[FIG1] Aerial view of the single-pixel CS camera in the lab [5].

Ref: Duarte et al, "Single pixel imaging via compressive sampling", IEEE Signal Processing Magazine, March 2008.

Rice Single Pixel Camera: a CS-based camera



Ref: Duarte et al, "Single pixel imaging via compressive sampling", IEEE Signal Processing Magazine, March 2008.

Rice Single Pixel Camera: a CS-based camera

- Contains no detector array.
- Light from the scene passes through Lens1 and is focussed on a digital micromirror device (DMD).
- DMD is a 2D array of thousands of very tiny mirrors.
- Light reflected from DMD passes through the second lens and to the photodiode.

Rice Single Pixel Camera: a CS-based camera

- The DMD acts as a random binary array of the same size as image \mathbf{f} . Each mirror in the DMD corresponds to one pixel in \mathbf{f} .
- A mirror can be either ON=1 (facing the Lens2) or OFF=0 (facing away from Lens2).
- The photodiode circuit acts as a photon counter – effectively, it measures the dot product between \mathbf{f} and ϕ_i , i.e. it measures:

$$y_i = \sum_{j=1}^n f_j \phi_{ij}$$

Rice Single Pixel Camera: a CS-based camera

- These values $\{y_i\}$ are output in the form of a voltage which is then digitized by an A/D converter.
- Note that a different binary code vector ϕ_i is used for each y_i , $1 \leq i \leq m$.
- The random binary code is implemented by setting the orientation of the mirrors (facing toward or away from Lens2) randomly within the hardware.
- One can implement +1,-1 random codes instead of 0,1 random codes by programming the diode to compute $2y_i - 1$ instead of y_i (equivalent to subtracting the mean light intensity from each measurement = measurement taken with all mirrors on).

Rice Single Pixel Camera: a CS-based camera

- The basic measurement model can be written as follows (in vector notation):

$$\mathbf{y} = \mathbf{\Phi}\mathbf{f}, \mathbf{\Phi} = [\boldsymbol{\varphi}_1 \mid \boldsymbol{\varphi}_2 \mid \dots \mid \boldsymbol{\varphi}_m]^T,$$

$$\mathbf{y} = (y_1, y_2, \dots, y_m)$$

- As per CS theory, there are guarantees of good reconstruction if the number of samples obeys (for K-sparse signals):

$$m \geq O(K \log(n / K))$$

Reconstruction Results



(a)



(b)



(c)

[FIG2] Single-pixel photo album. (a) 256×256 conventional image of a black-and-white R. (b) Single-pixel camera reconstructed image from $M = 1,300$ random measurements ($50\times$ sub-Nyquist). (c) 256×256 pixel color reconstruction of a printout of the Mandrill test image imaged in a low-light setting using a single photomultiplier tube sensor, RGB color filters, and $M = 6,500$ random measurements.

Optimization technique used:

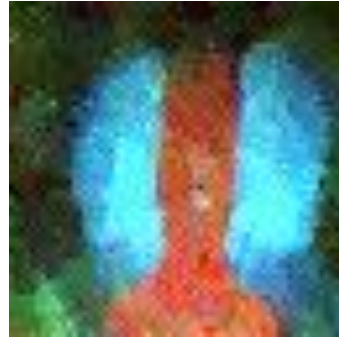
$$\min_f TV(f) \text{ such that } y = \Phi f$$

More Results:

<http://dsp.rice.edu/cscamera>



Original



4096 pixels,
800
measurements,
i.e. 20% data



65536 pixels, 6600 measurements, i.e. 10% data

Informal description of Rice Single Pixel Camera:

<http://terrytao.wordpress.com/2007/04/13/compressed-sensing-and-single-pixel-cameras/>

Compressed sensing on the chip

- This is a compressive camera developed at Stanford, that uses the same mathematical model as the Rice SPC.
- The difference is that it calculates all the m dot products on a single CMOS chip.
- What dot products? Of a random pattern (with n elements) with a vector of n analog pixel values.
- Only the $m \ll n$ dot products are quantized (Analog to digital conversion), saving huge amounts of energy.
- Mounted on a mobile phone – led to 15 fold savings in battery power.
- See [here](#) for more information.
- Yields excellent quality reconstruction with high frame rates (960 fps).

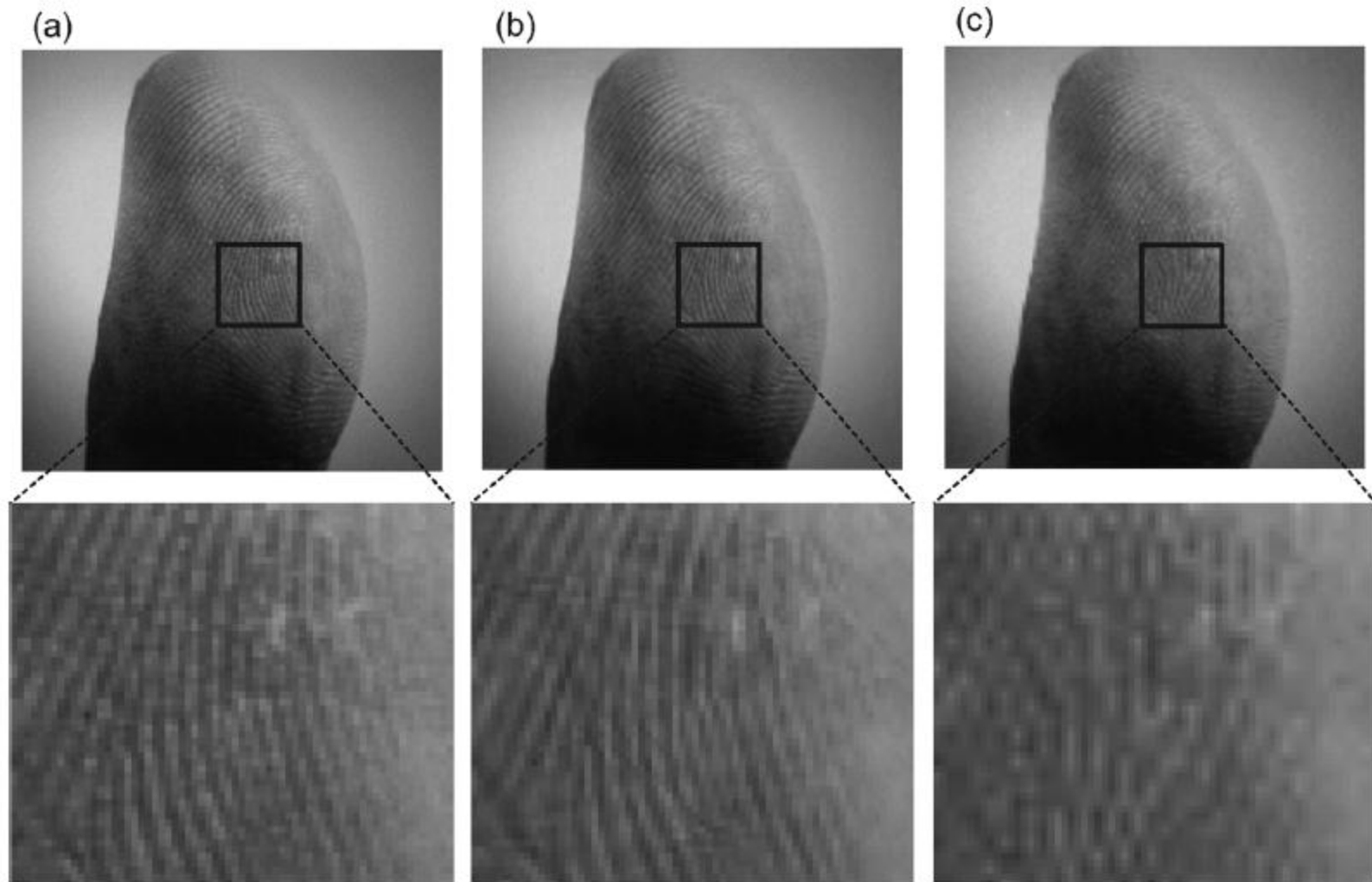


Fig. 16. Sample images captured in: (a) normal mode at 120 fps, (b) compressed sensing at $CR = 1/4$ and 480 fps, (c) downsampling at 1/4 ratio.

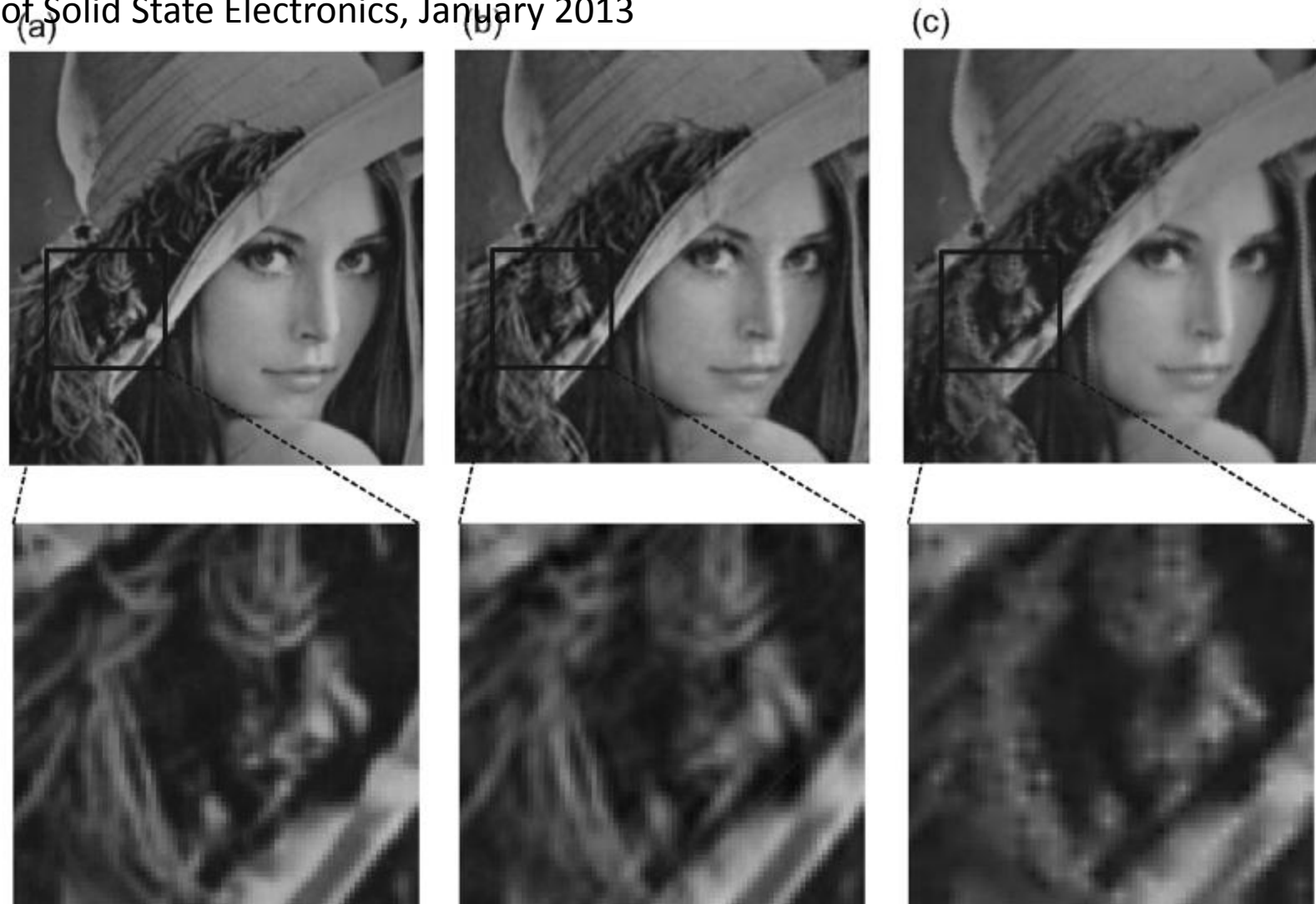


Fig. 17. Sample images captured in: (a) normal mode at 120 fps, (b) compressed sensing at $CR = 1/8$ and 960 fps, (c) downsampling at 1/8 ratio.

PART 2: RICE SINGLE PIXEL CAMERA FOR VIDEO ACQUISITION

Streaming Video Acquisition

- SPC can be extended for video.
- Consider a video with a total of F (2D) images.
- In the still-image SPC, an image was coded several times using different binary codes $\phi_{i,t}$.
- Note that in a video-camera, this reduces the **video frame rate**.
- Assume we take a total of M measurements, i.e. M/F measurements (**snapshots**) per frame.
- We make the simplifying assumption that the **scene changes slowly** within the set of M/F snapshots.

Streaming Video Reconstruction

- **Method 1:** To reconstruct the original video from the CS measurements, we could use a 2D DCT/wavelet basis Ψ and perform F independent (2D) frame-by-frame reconstructions, by solving:

$$\forall t \in \{1, \dots, F\}, \min_{\theta_t} \|\theta_t\|_1 \text{ such that } \mathbf{y}_t = \Phi_t \mathbf{f}_t = \Phi_t \Psi \theta_t, \\ \Phi_t \in R^{M/F \times n}, \Psi \in R^{n \times n}, \theta_t \in R^n, \mathbf{y}_t \in R^{M/F}$$

- This procedure fails to exploit the tremendous **inter-frame redundancy** in natural videos.

Streaming Video Acquisition

- **Method 2:** Create a joint measurement matrix Φ for the entire video sequence, as shown below. Φ is block-diagonal, with each of the diagonal blocks being the matrix Φ_t for measurement \mathbf{y}_t .

$$\Phi = \begin{pmatrix} \Phi_1 & \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \Phi_2 & & \mathbf{0} \\ & & \ddots & \\ \mathbf{0} & & & \Phi_F \end{pmatrix}, \Phi \in \mathcal{R}^{M \times Fn}, \Phi_i \in \mathcal{R}^{M/F \times n}$$

$$\mathbf{y} = (\mathbf{y}_1 | \mathbf{y}_2 | \dots | \mathbf{y}_F), \mathbf{y}_i = \Phi_i \mathbf{f}_i$$

Streaming Video Acquisition

- **Method 2 (continued)** : Use a 3D DCT/wavelet basis Ψ (size Fn by Fn) for sparse representation of the video sequence:

$$\min_{\theta} \|\theta\|_1 \text{ such that } \mathbf{y} = \Phi \mathbf{f} = \Phi \Psi \theta,$$

$$\Phi \in R^{M \times Fn}, \Psi \in R^{Fn \times Fn}, \theta \in R^{Fn}, \mathbf{y} \in R^M$$

Streaming Video Acquisition

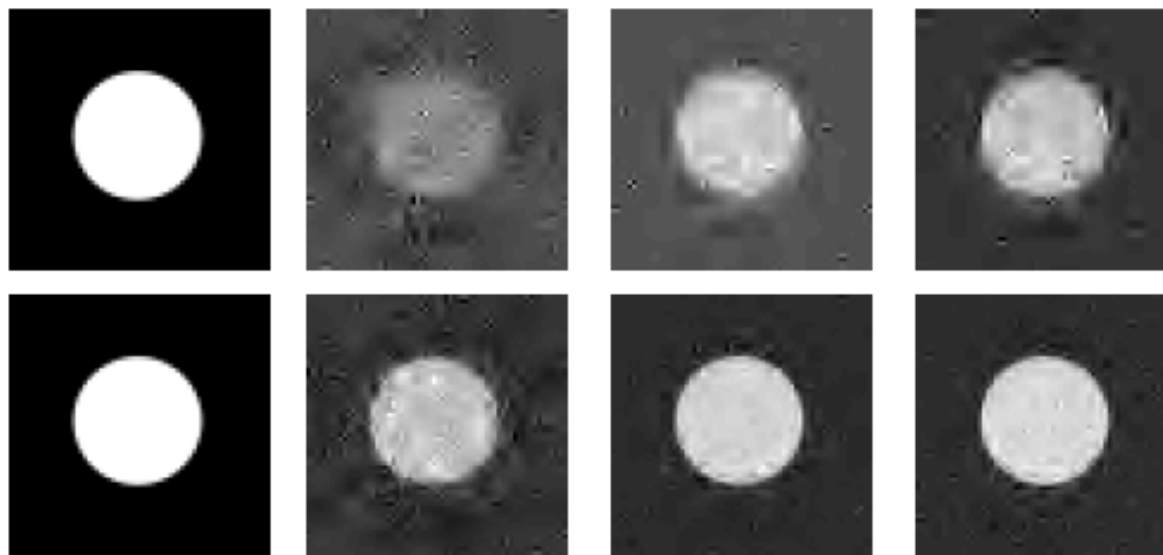
- **Method 3 (Hypothetical):** Assume we had a 3D SPC with a full 3D sensing matrix Φ which operates on the full video, and with an associated 3D wavelet basis.

$$\min_{\theta} \|\theta\|_1 \text{ such that } \mathbf{y} = \Phi \mathbf{f} = \Phi \Psi \theta,$$

$$\Phi \in R^{M \times Fn}, \Psi \in R^{Fn \times Fn}, \theta \in R^{Fn}, \mathbf{y} \in R^M$$

Results

- Experiment performed on a video of a moving disk (against a constant background) - size 64×64 with $F = 64$ frames.
- This video is sensed with a total of M measurements with M/F measurements per frame.
- All three methods (frame-by-frame 2D, 2D measurements with 3D reconstruction, 3D measurements with 3D reconstruction) compared for $M = 20000$ and $M = 50000$.



Source of images:
Duarte et al, "Compressive
imaging for video
representation and
coding",
http://www.ecs.umass.edu/~mduarte/images/CSCamera_PCS.pdf

(a) frame 32 (b) 2D meas 2D recon (c) 2D meas 3D recon (d) 3D meas 3D recon

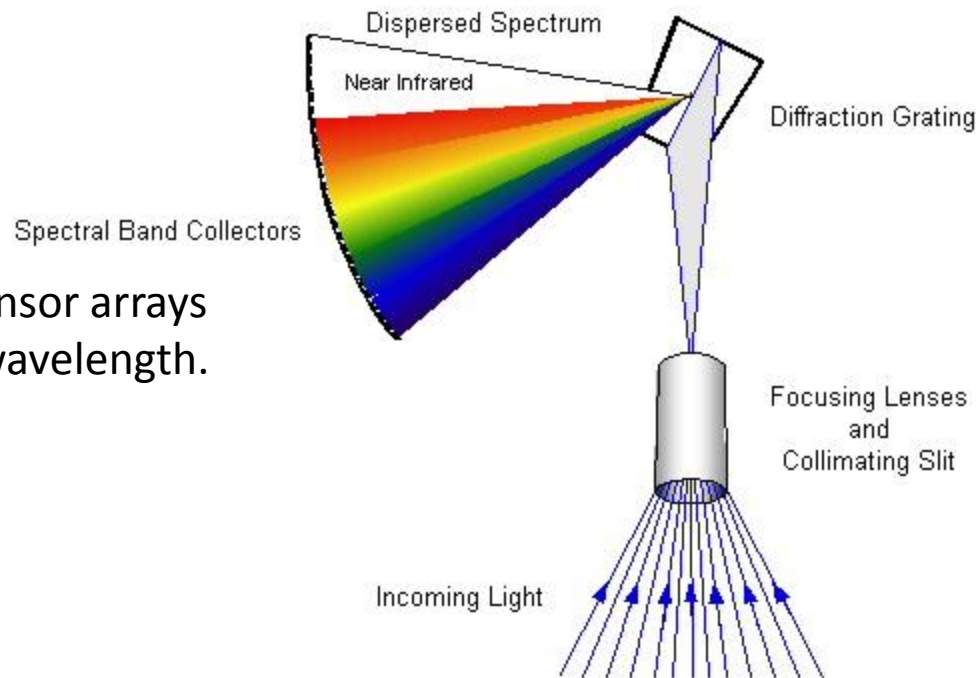
Fig. 3. Frame 32 from reconstructed video sequence using (top row) $M = 20,000$ and (bottom row) $M = 50,000$ measurements. (a) Original frame. (b) Frame-by-frame 2D measurements; frame-by-frame 2D reconstruction; $MSE = 3.63$ and 0.82 . (c) Frame-by-frame 2D measurements; joint 3D reconstruction; $MSE = 0.99$ and 0.24 . (d) Joint 3D measurements; joint 3D reconstruction; $MSE = 0.76$ and 0.18 . The results in (d) are comparable to the MSE obtained by wavelet thresholding with $K = 655$ and 4000 coefficients, respectively.

Recent developments

- A recent development in video reconstruction with SPC framework: linear dynamical system model for video (we will not cover it in this course).
- Provides appealing results for even 100x compression rates (simulated results).
- <http://www.ece.rice.edu/~as48/research/cslds/>

PART 3: COMPRESSIVE ACQUISITION OF HYPERSPECTRAL DATA

Conventional Hyperspectral Camera



Multiple sensor arrays
– one per wavelength.
Expensive!

CASSI: Compressive Hyperspectral Image Acquisition

- Reconstruction of hyperspectral data imaged by a coded aperture snapshot spectral imager (CASSI) developed at the DISP (Digital Imaging and Spectroscopy) Lab at Duke University.
- CASSI measurements are a superposition of aperture-coded wavelength-dependent data: ambient **3D hyperspectral datacube** is mapped to a **2D 'snapshot'**.
- **Task:** Given one or more 2D snapshots of a scene, recover the original scene (3D datacube).

CASSI: Description of Measured Data

Ref: A. Wagadarikar et al, "Single disperser design for coded aperture snapshot spectral imaging", Applied Optics 2008.

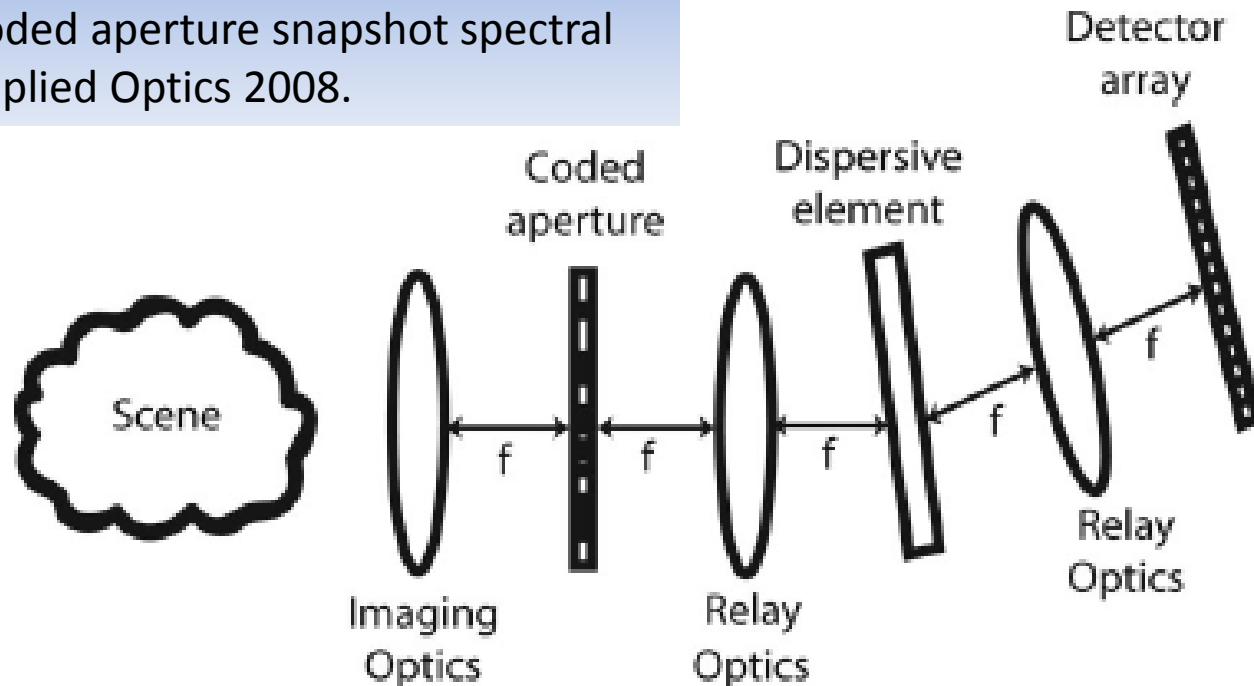
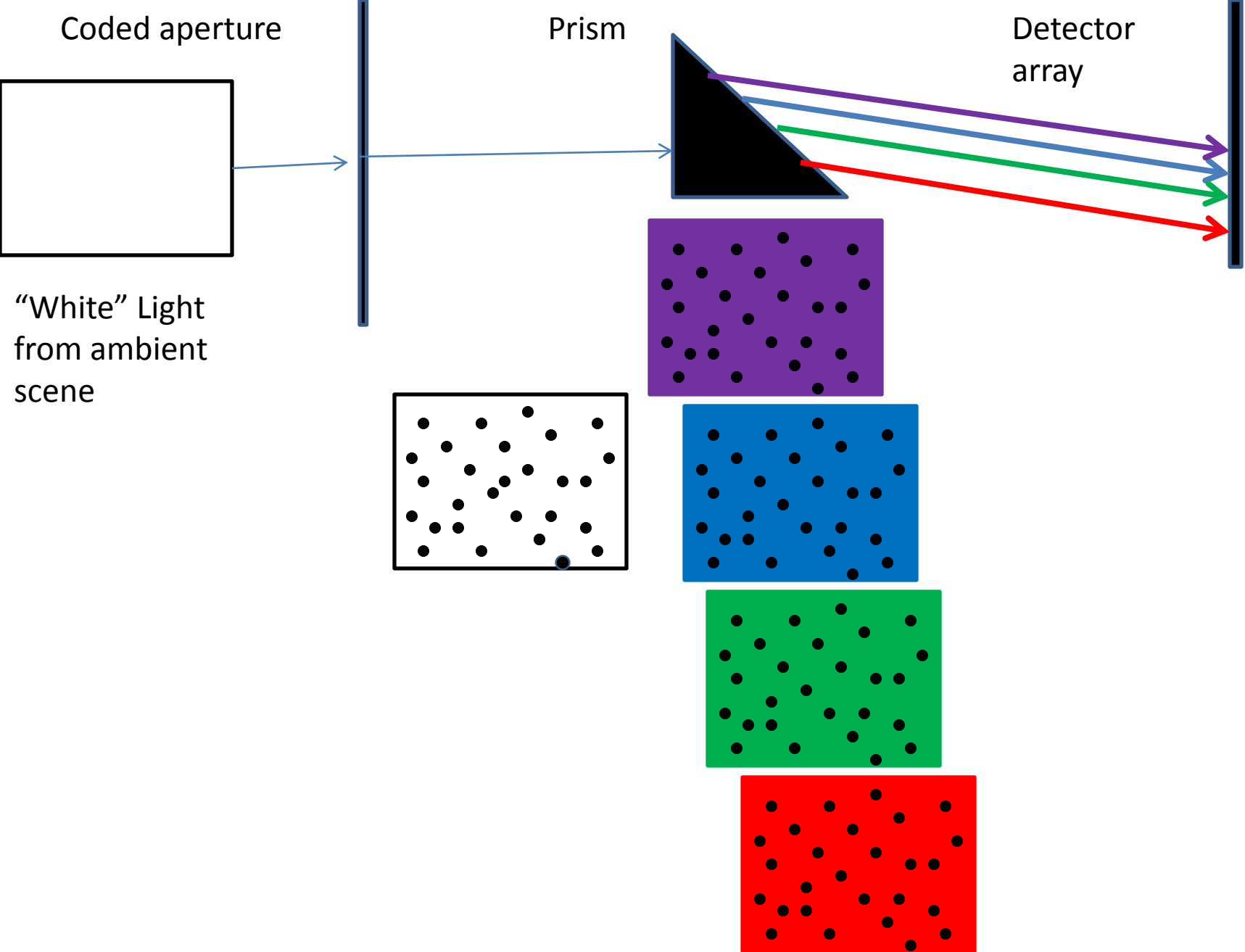


Fig. 1. Schematic of a SD CASSI. The imaging optics image the scene onto the coded aperture. The relay optics relay the image from the plane of the coded aperture to the detector through the dispersive element.



CASSI: Description of Measured Data

- Assume we want to measure a hyperspectral data-cube given as $\mathbf{X} \in R^{N_x \times N_y \times N_\lambda}$ where data at each wavelength is a 2D image of size $N_x \times N_y$ where the number of wavelength is N_λ .
- In a CASSI camera, each image $\mathbf{X}_j, 1 \leq j \leq N_\lambda$, is multiplied by the same known (random) binary code given as $\mathbf{C} \in \{0,1\}^{N_x \times N_y}$ yielding an image

$$\hat{\mathbf{X}}_j = \mathbf{X}_j \bullet \mathbf{C}.$$

CASSI: Description of Measured Data

- Let the pixel at location (x, y) in image $\hat{\mathbf{X}}_j$ be denoted as $\hat{X}_j(x, y)$. The shifted version of $\hat{X}_j(u, v)$ is given as $S_j(x, y) = \hat{X}_j(x - l_j, y)$ where $l_j > 0$ denotes the shift in the pixels at wavelength $\lambda_j, l_j \neq l_{\hat{j}}, j \neq \hat{j}$.
- The wavelength-dependent shifts are implemented by means of a **prism** in the CASSI camera, whereas modulation by the binary code is implemented by means of a **mask**.

CASSI: Description of Measured Data

- The measurement by the CASSI system is a single 2D “snapshot” given as follows (superposition of coded data from all wavelengths):

$$M(x, y) = \sum_{j=1}^{N_\lambda} S_j(x, y) = \sum_{j=1}^{N_\lambda} \hat{X}_j(x - l_j, y) = \sum_{j=1}^{N_\lambda} X_j(x - l_j, y) \bullet C(x - l_j, y)$$

- Due to the wavelength-dependent shifts, the contribution to $M(x, y)$ at different wavelengths corresponds to a different spatial location in each of the slices of the datacube \mathbf{X} .
- Also the portions of the coded aperture contributing towards a single pixel value $M(x, y)$ are different for different wavelengths.

Multi-frame CASSI

- The compression rate of CASSI is the number of wavelengths: 1.
- This compression rate can be reduced if $T > 1$ snapshots of the same scene are acquired in quick succession, denoted as $\{\mathbf{M}_t\}_{t=1}^T$ reducing the compression rate to $N_\lambda : T$.
- Each snapshot is acquired using a **different aperture code**, i.e. a **different mask pattern** - implemented in hardware by **moving the position of the mask** using a piezo-electric mechanism.
- Reduction in compression rate = **less ill-posed problem** = scope for better reconstruction.

Ref: D. Kittle et al, "Multiframe image estimation for coded aperture snapshot spectral imagers", Applied Optics, 2010.

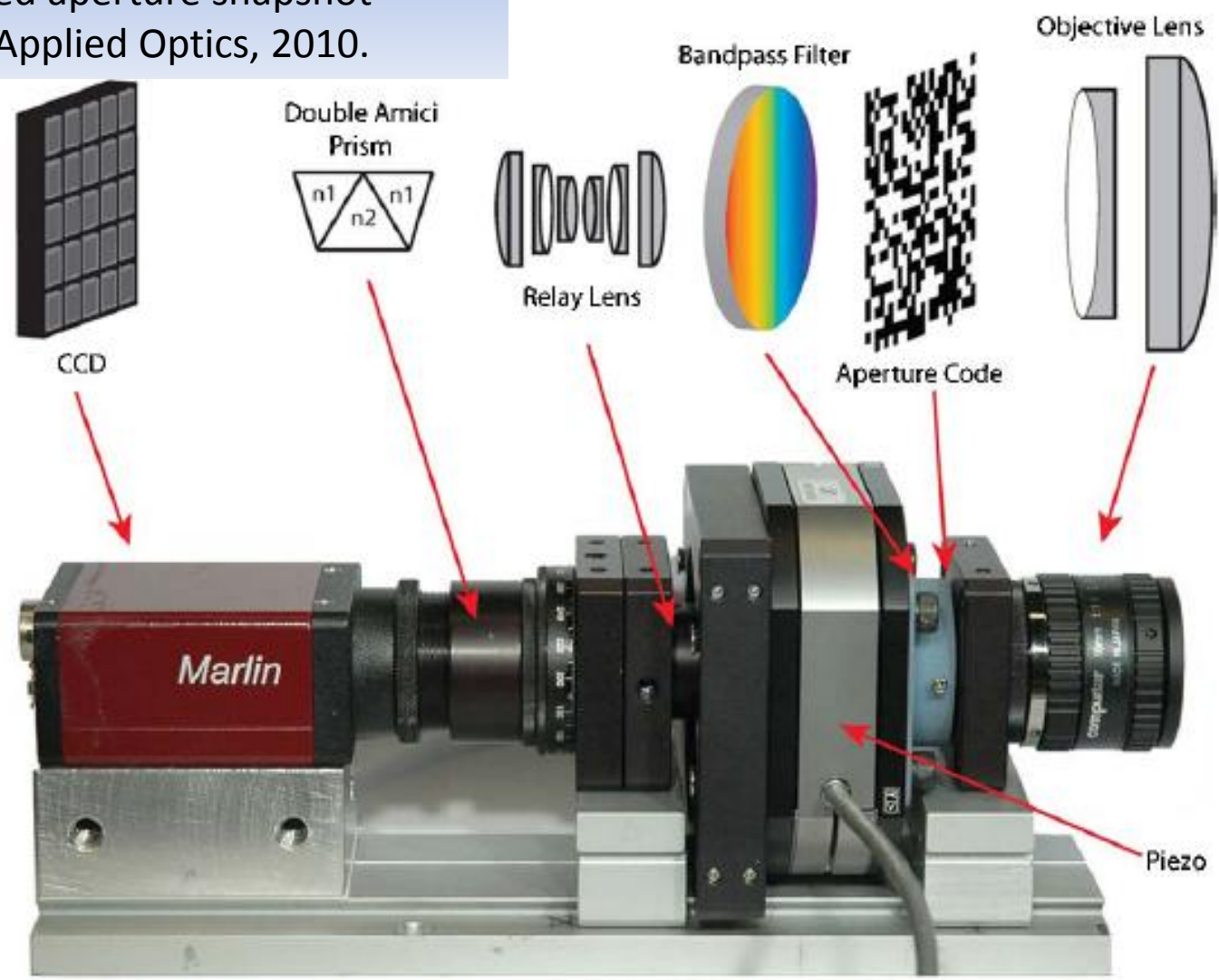


Fig. 2. (Color online) Schematic and photo showing the entire CASSI instrument. Left to right: CCD, double Amici prism, relay lens, piezo stage, bandpass filter, aperture code, and objective lens.

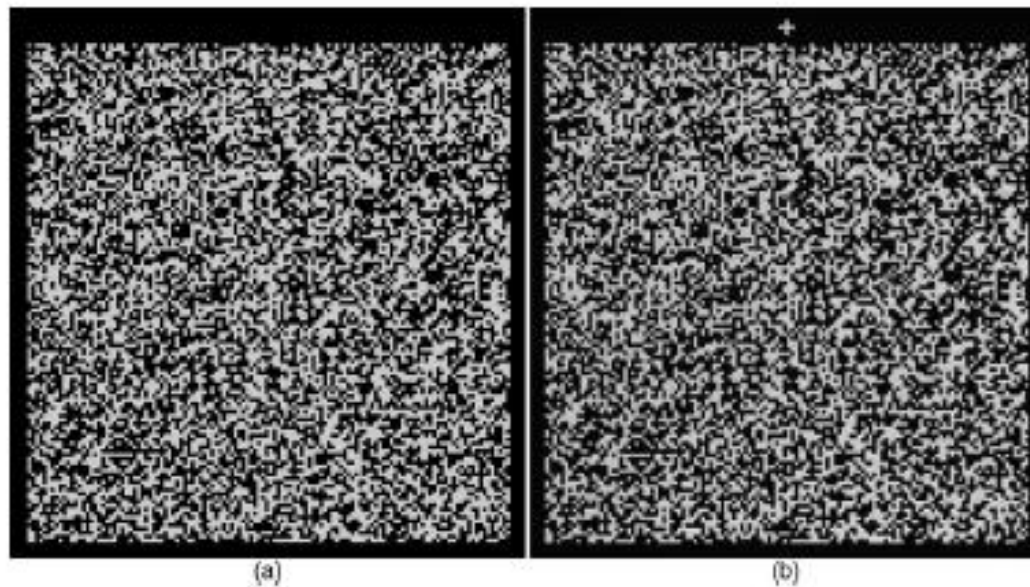
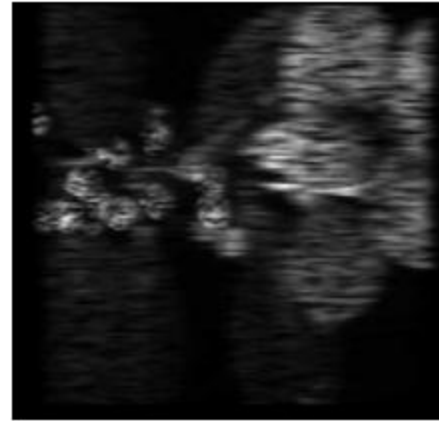


Fig. 3. (a) Ideal aperture code and (b) image of aperture code at detector under 550 nm monochromatic light. Notice the blur added to the aperture code relative to the ideal code.

- Aperture Code: created randomly (random binary, $[0,1]$ uniform random also possible)
- The aperture code pattern has holes of size 2×2 pixels (smaller holes give rise to diffraction artifacts).
- The pattern is projected onto the detector array in a magnified form.
- Note: Random mask pattern is needed as per CS theory.

Sample CASSI Measurements versus RGB representation of underlying hyperspectral scene



Size 1000 x 700 x 24



Size 440 x 440 x 23


Reconstruction Method

- A total-variation based CS solver called as TwIST was used (ref: Bioucas-Dias and Figueredo, A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration”, *IEEE Transactions on Image Processing*, 2007.)
- The inversion is performed by solving the following:

$$E(\mathbf{f}^*) = \min_f \sum_t \|\mathbf{m}_t - \Phi_t \mathbf{f}\|^2 + \tau TV(\mathbf{f}),$$

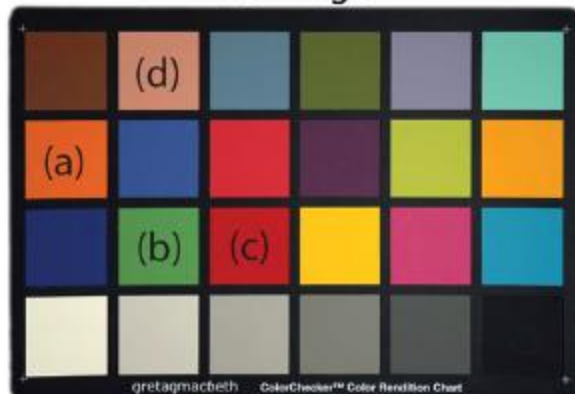
$$TV(\mathbf{f}) = \sum_{x=1}^{N_x} \sum_{y=1}^{N_y} \sum_{z=1}^{N_\lambda} \sqrt{(f(x+1, y, \lambda) - f(x, y, \lambda))^2 + (f(x, y+1, \lambda) - f(x, y, \lambda))^2}$$

Reconstruction Method

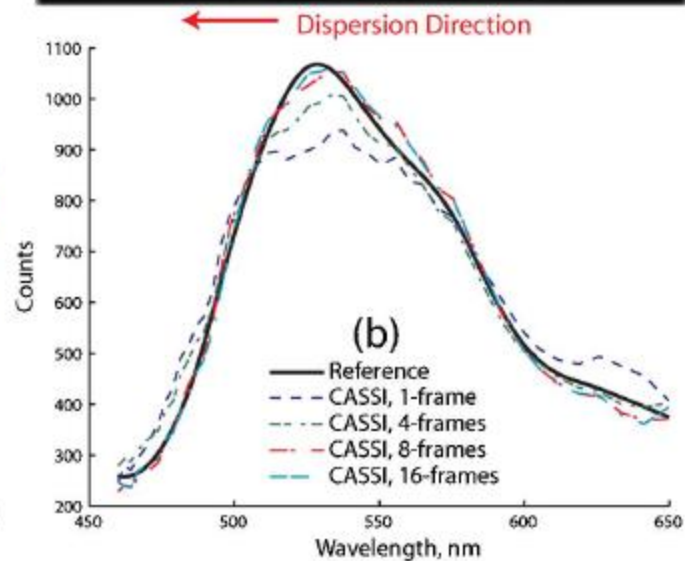
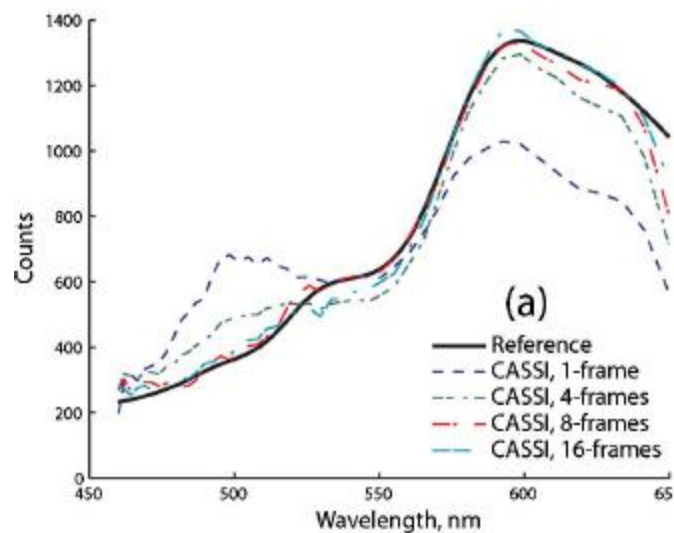
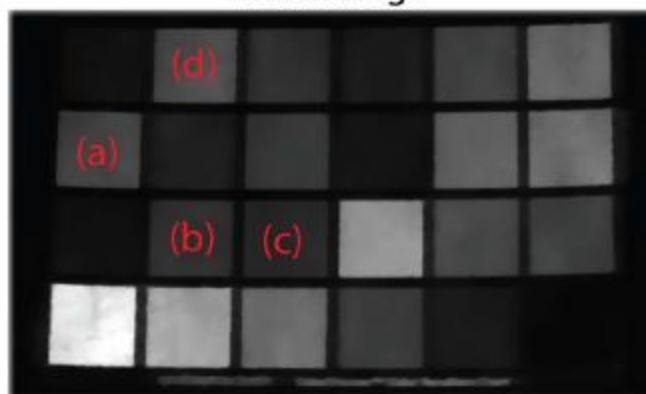
$$E(\mathbf{f}^*) = \min_f \sum_t \|\mathbf{m}_t - \Phi_t \mathbf{f}\|^2 + \tau TV(\mathbf{f}),$$


Known forward model (sensing matrix) for the t -th snapshot measurement, i.e. \mathbf{m}_t (governed by several factors – the exact aperture code and its position relative to the scene, plus any blurring effects due to the hardware)

RGB Image



CASSI Image



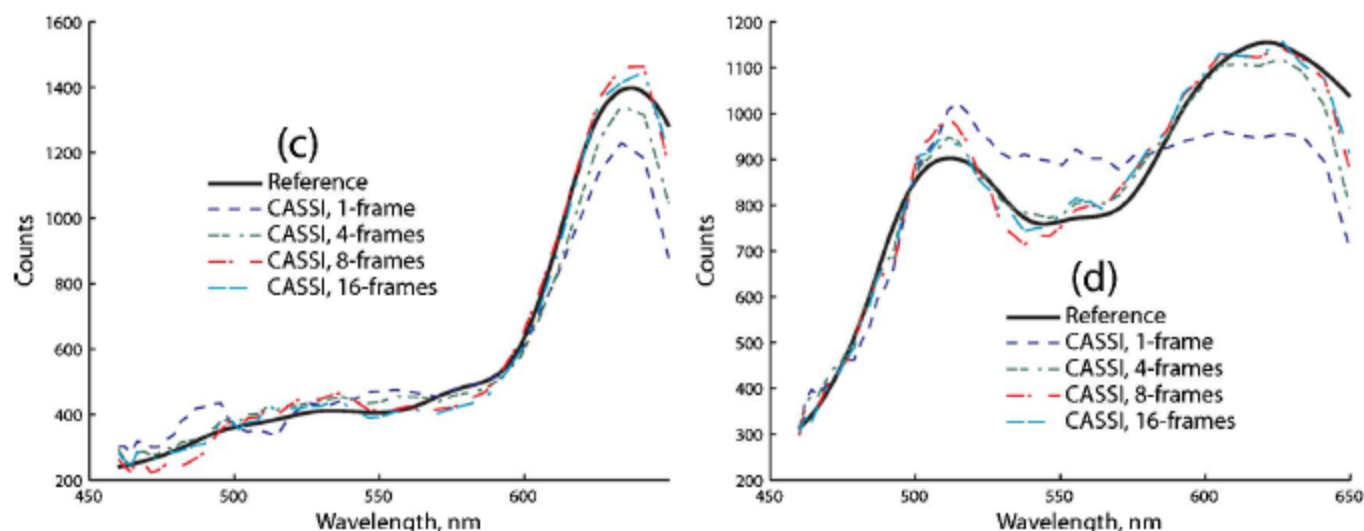


Fig. 5. (Color online) Using the GretagMacbeth ColorChecker as a baseline, comparisons were made with white-light sunlight emulation. (a)–(d) compare snapshot versus 4, 8, and 16 frame reconstructions. Nearby colors can bleed into the spectra, as seen in the RGB image, where the blue just to the right of the orange square was dispersed into the orange. Multiple frames eliminate this problem.

Table 2. RMSE of CASSI Spectra versus Reference Spectrometer

Frames	Center				Edge			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
1	9.6%	6.9%	4.1%	17.4%	14.0%	11.7%	8.4%	34.3%
4	2.7%	3.6%	2.4%	4.5%	5.3%	8.1%	2.7%	10.2%
8	3.7%	3.3%	2.6%	5.4%	2.7%	3.6%	2.3%	8.9%
16	1.9%	3.4%	2.8%	5.0%	2.9%	4.1%	2.7%	6.8%

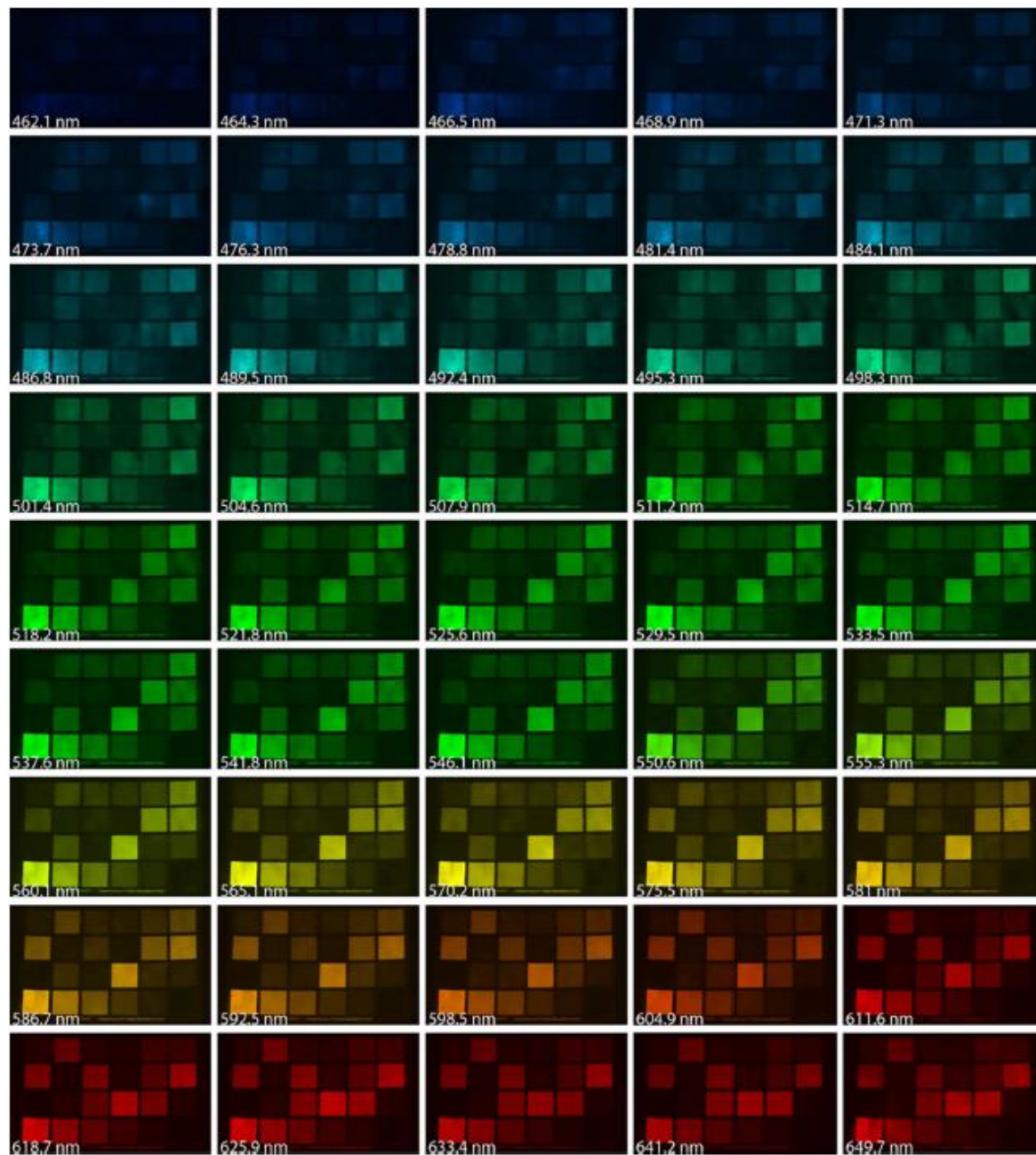


Fig. 6. (Color online) Multiframe CASSI reconstruction of the GretagMacbeth ColorChecker, showing all the channels from 460–650 nm.

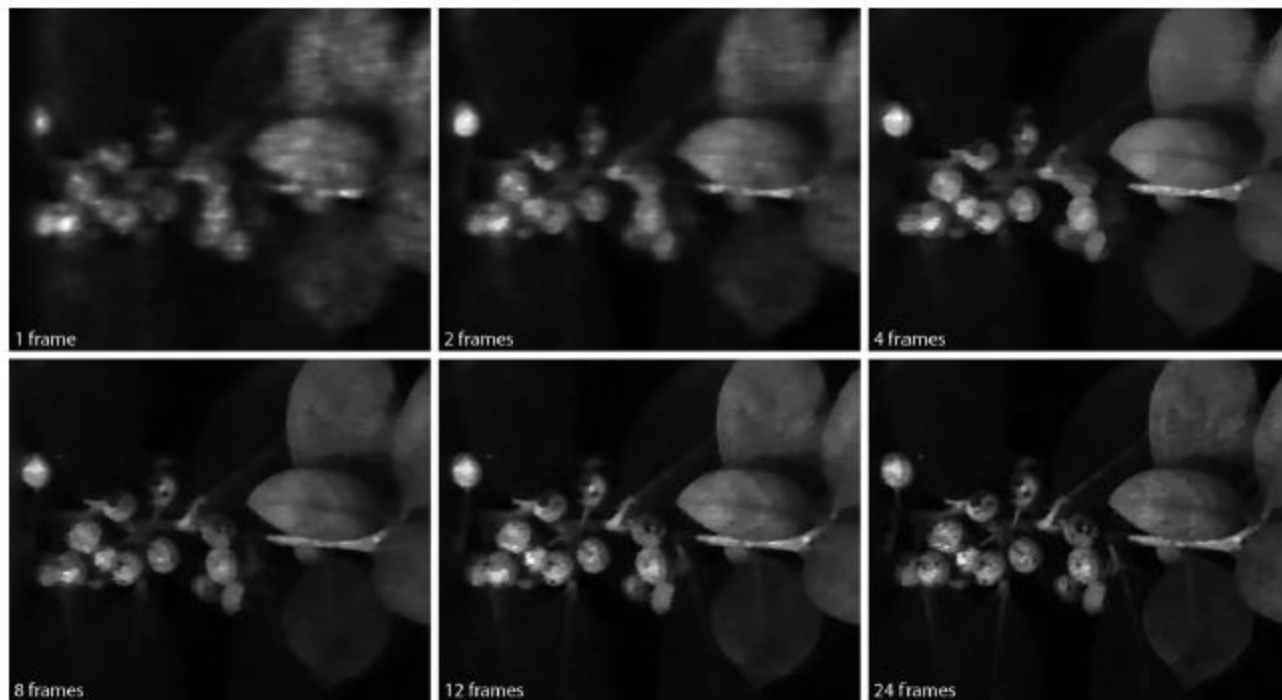


Fig. 7. Comparison between a single snapshot and various multiframe reconstructions for a single channel at 605 nm.

**Table 3. PSNR versus Number of CASSI
Frames for Actual Data, Referenced
to a 24-Frame Image in Fig. 7**

Frames	PSNR
1	23.2
2	27.3
4	29.0
8	30.7
12	35.7

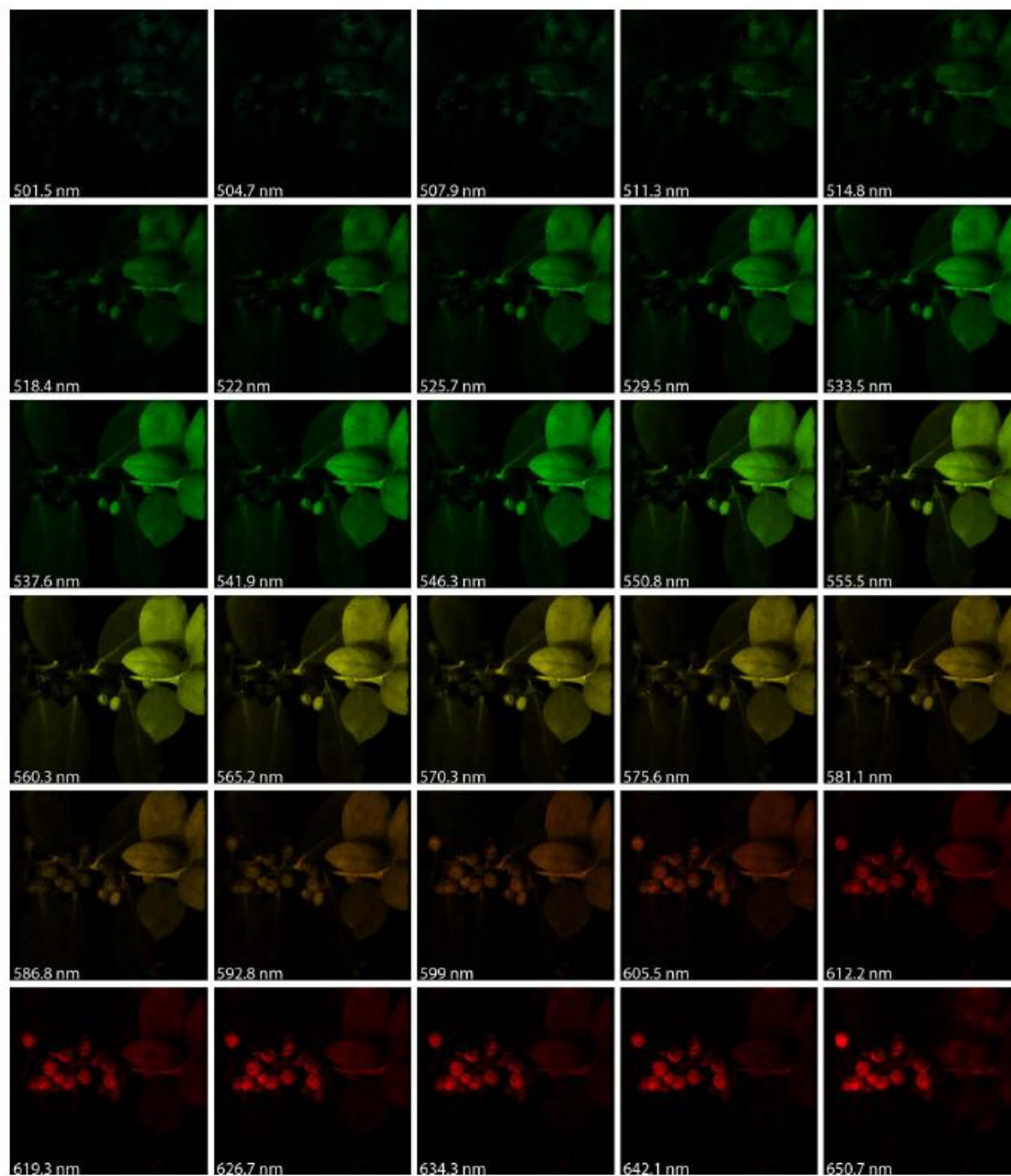


Fig. 8. (Color online) All channels from 500–650 nm from a 24-frame reconstruction of the data cube.

PART 4: COMPRESSIVE VIDEO ACQUISITION USING CODED SNAPSHOTS

Video from a Single Coded Exposure Photograph using a Learned Over- Complete Dictionary

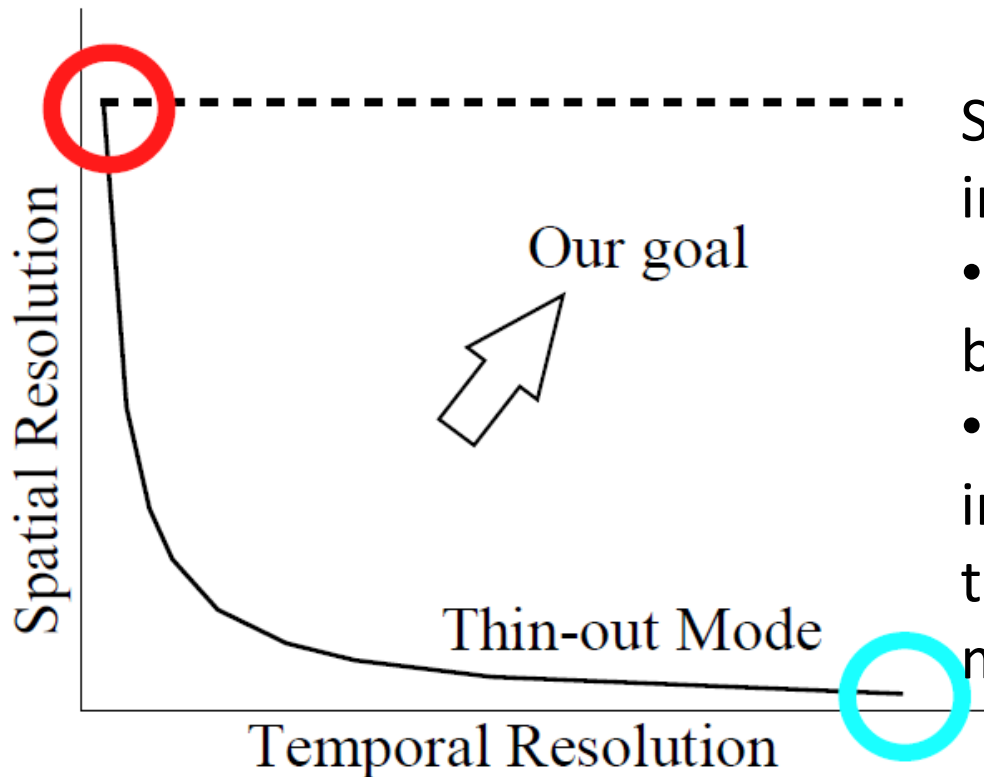
Authors: Yasunobu Hitomi, Jinwei Gu,
Mohit Gupta, Tomoo Mitsunaga,
Shree Nayar

Published in ICCV 2011

Basic Goal of the Paper

- Application of “computational photography”
- Improve frame rate of a video camera by making appropriate changes to hardware WITHOUT sacrificing spatial resolution.

Space-Time Tradeoff



Sampling every k -th row of an image frame:

- Spatial resolution decreases by factor of k ,
- Temporal resolution increases by factor of k (for the same number of measurements)

Can be overcome with more sophisticated hardware –
but associated cost is HIGH



(b) Motion blurred image



Still camera



(c) Thin-out mode: Low spatial resolution, high frame rate



(d) Our input: A single coded exposure image



(e) Our result: High spatial resolution, high frame rate video

Coded Exposure Image

It is a coded superposition (i.e. summation) of N sub-frames within a unit integration time of the video camera.

Coded exposure image
(captured in one unit
integration time of the
camera)

$$I(x, y) = \sum_{t=1}^N S(x, y, t) \cdot E(x, y, t).$$

Binary code at
time instant t

Sub-frame at
time instant t

Conventional capture
(simple integration across
time, without modulation
by binary codes)



$$S(x, y, t) = 1, \forall(x, y, t)$$



(b) Motion blurred image



Still camera



(c) Thin-out mode: Low spatial resolution, high frame rate



$I(x,y)$

(d) Our input: A single coded exposure image



(e) Our result: High spatial resolution, high frame rate video

$E(:, :, 1), t=1$

$E(:, :, 18), t=18$

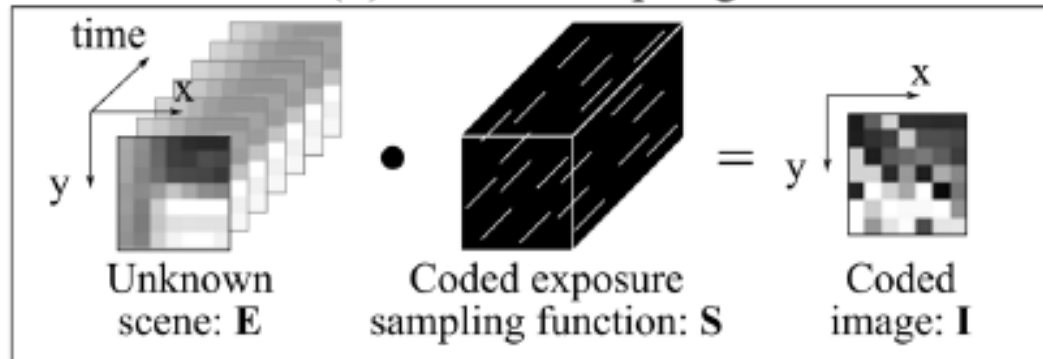
$E(:, :, 36), t=36$

Explanation

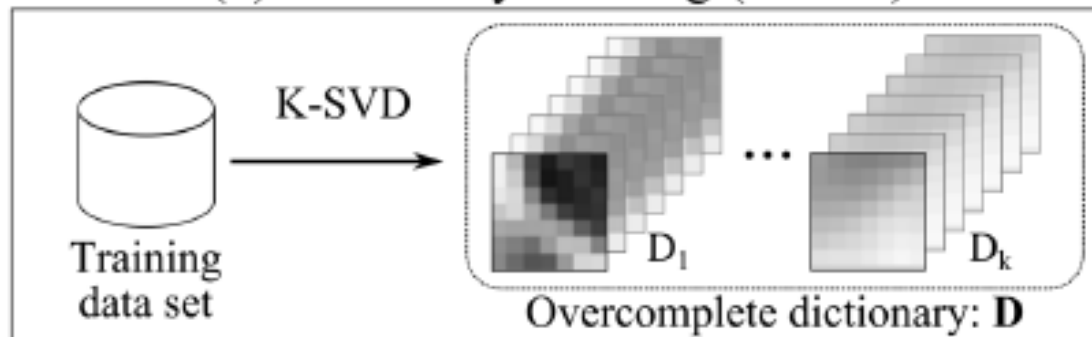
- Imagine a **30 fps** off the shelf video-camera. It acquires one frame in **1/30 seconds** (this is the **unit integration time of the video camera**).
- The camera model in this paper will acquire a **coded exposure image** $I(x,y)$ in the **same** amount of time.
- From this coded exposure image, we will be able to **reconstruct $N = 20$ sub-frames** (all showing changes that occurred in the scene **within the 1/30 seconds period**), using a standard Compressive Sensing Reconstruction algorithm.
- Thus we are doing **20-fold temporal super-resolution**, and that too **without sacrificing spatial resolution**.
- Effectively we are increasing the camera frame rate from 30 fps to $30 \times 20 = 600$ fps!

Dictionary Learning/Sparse Coding

(1) Coded sampling



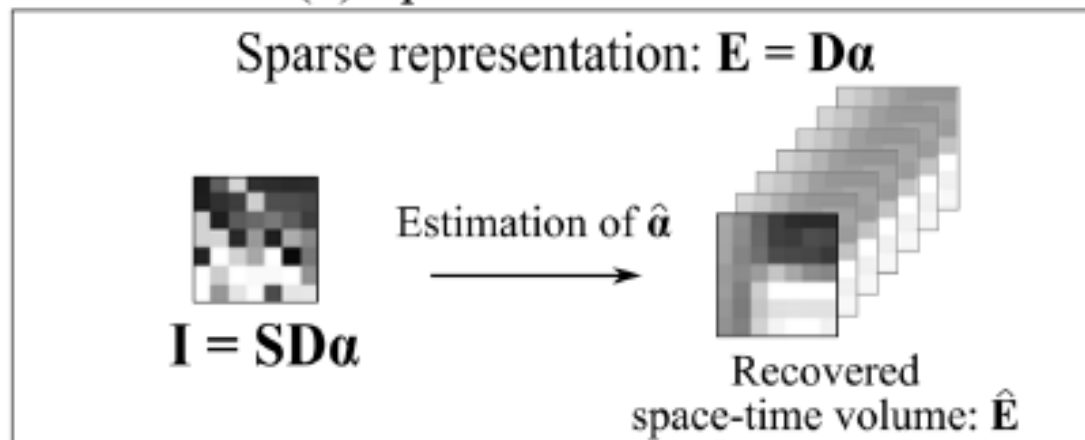
(2) Dictionary learning (offline)



$$\mathbf{E} = \mathbf{D}\alpha = \alpha_1\mathbf{D}_1 + \cdots + \alpha_k\mathbf{D}_k.$$

Dictionary Learning/Sparse Coding

(3) Sparse reconstruction

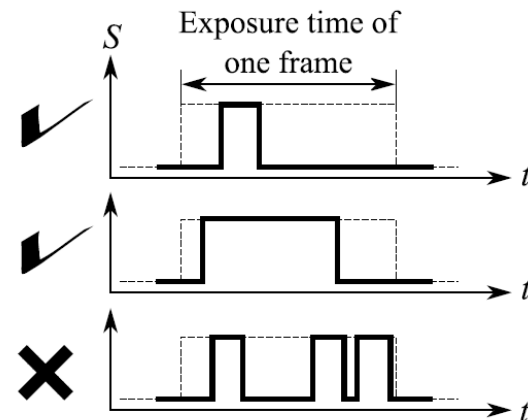
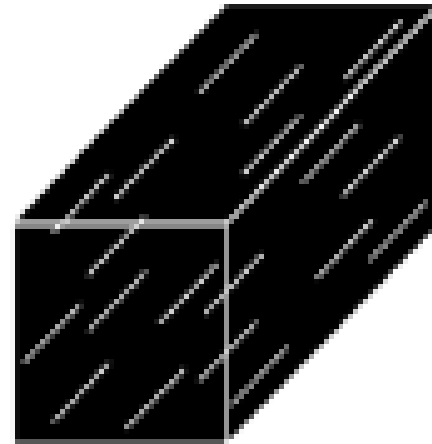


$$\mathbf{E} = \mathbf{D}\boldsymbol{\alpha} = \alpha_1 \mathbf{D}_1 + \cdots + \alpha_k \mathbf{D}_k.$$

$$\hat{\boldsymbol{\alpha}} = \arg \min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha}\|_0 \quad \text{subject to} \quad \|\mathbf{S}\mathbf{D}\boldsymbol{\alpha} - \mathbf{I}\|_2^2 < \epsilon.$$

Code Design: S

- (1) S should be binary – at any point of time, a pixel (that collects light) is **either ON or OFF**
- (2) Each pixel can have only one **continuous ON time** (called a **‘bump’**) during the camera integration time (due to limitations of contemporary CMOS sensors)
- (3) **Fixed bump length** for all pixels – but **different start times** for the bump at different pixels
- (4) Union of bumps within an $M \times M$ spatial patch should **cover full integration time**



Optimal Bump Length?

Too high: removes high frequency information

Too low: low SNR

Bump length	Noise standard deviation σ (Grey-levels)					
	0	1	4	8	15	40
1	22.96	22.93	22.88	22.50	21.41	17.92
2	23.23	23.22	23.18	23.06	22.62	20.76
3	23.37	23.37	23.35	23.25	23.03	21.69
4	23.29	23.30	23.25	23.27	22.99	22.08
5	23.25	23.26	23.24	23.19	23.07	22.34
6	23.06	23.10	23.07	23.06	22.85	22.32
7	22.93	22.92	22.89	22.85	22.80	22.29
8	22.80	22.81	22.77	22.78	22.69	22.23
9	22.63	22.62	22.61	22.59	22.53	22.09
10	22.49	22.48	22.50	22.49	22.43	22.06

Set to 2 for 9X frame rate gain, to 3 for 18X frame rate gain

Dictionary Learning

- Done offline – training set was 20 video sequences, each video rotated in 8 directions and played forward + backward = 320 videos.
- All videos had target frame rate (500 to 1000 fps, as we work with a 60 fps camera and want 9-18 fold gain).
- Video-patch size was $7 \times 7 \times 36 = 1764 \times 1$
- Offline learning: KSVD (*), $K = 100,000$ atoms
- Sparse coding done online (using OMP)

KSVD is a dictionary learning technique. Given a set of input patches in $n \times N$, it learns a set of vectors ($n \times K$), such that a sparse linear combination of these vectors approximates each patch as closely as possible. There are K coefficients per patch, out of which most are encouraged to be 0 (sparse).

Results on toy-data



3D DCT (1764 bases)
PSNR = 21.95



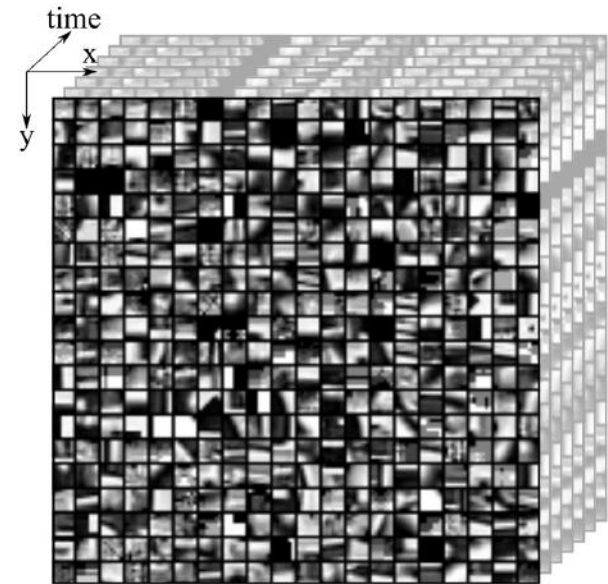
3D DWT (1764 bases)
PSNR = 15.78



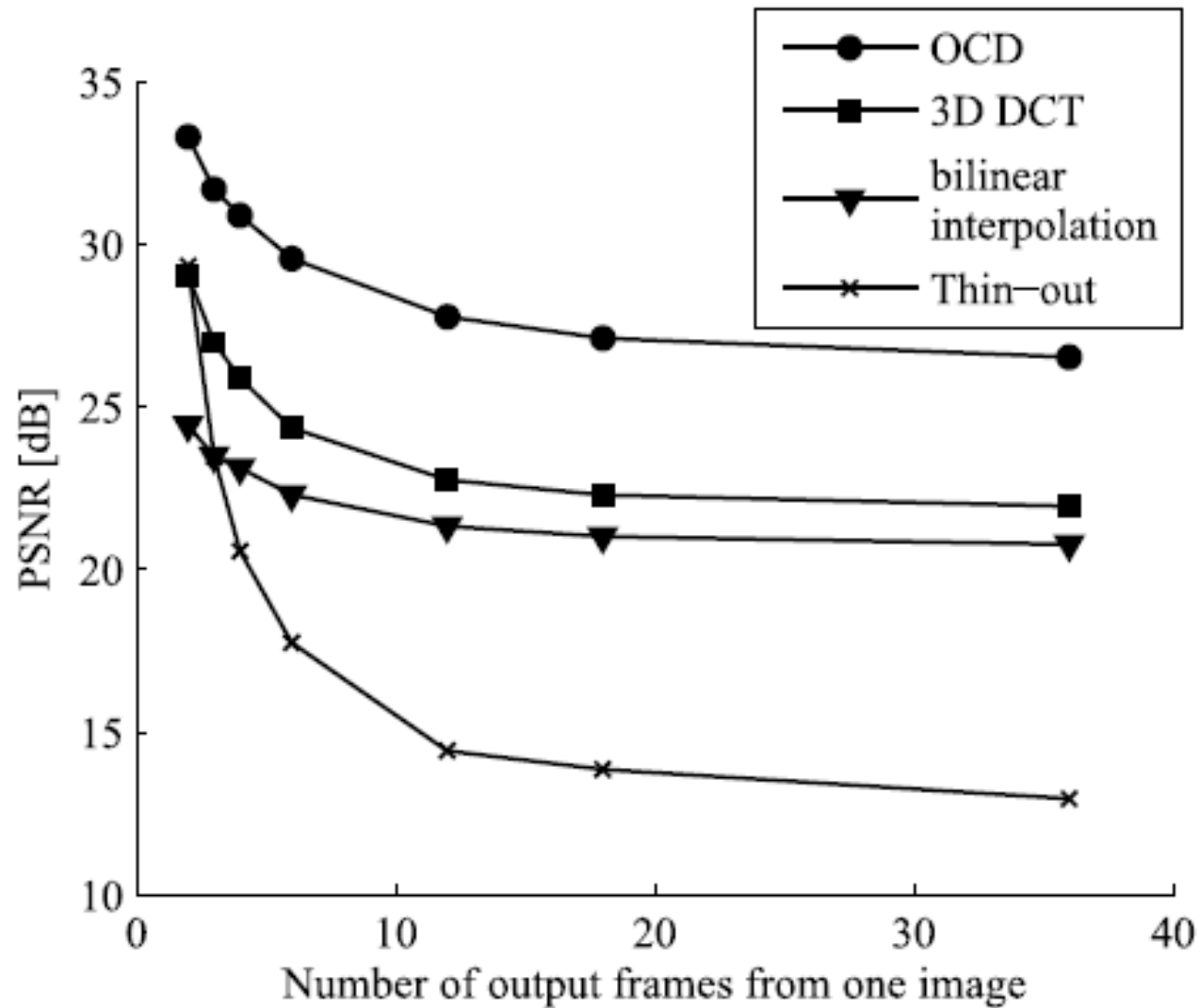
Learned Dictionary (1764 bases)
PSNR = 24.21



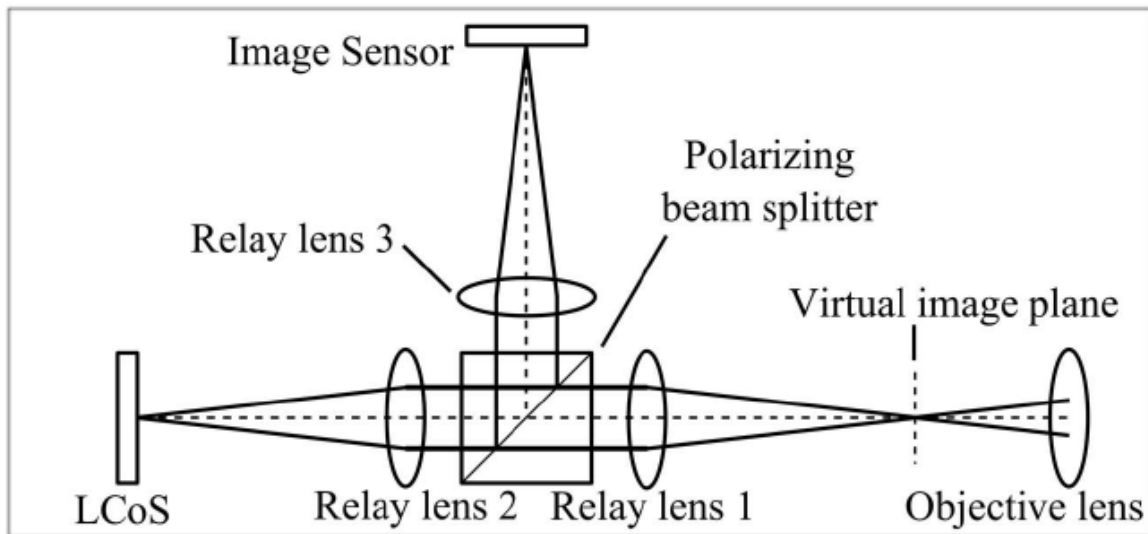
Learned Dictionary (100k bases)
PSNR = 26.54



36X frame
rate gain

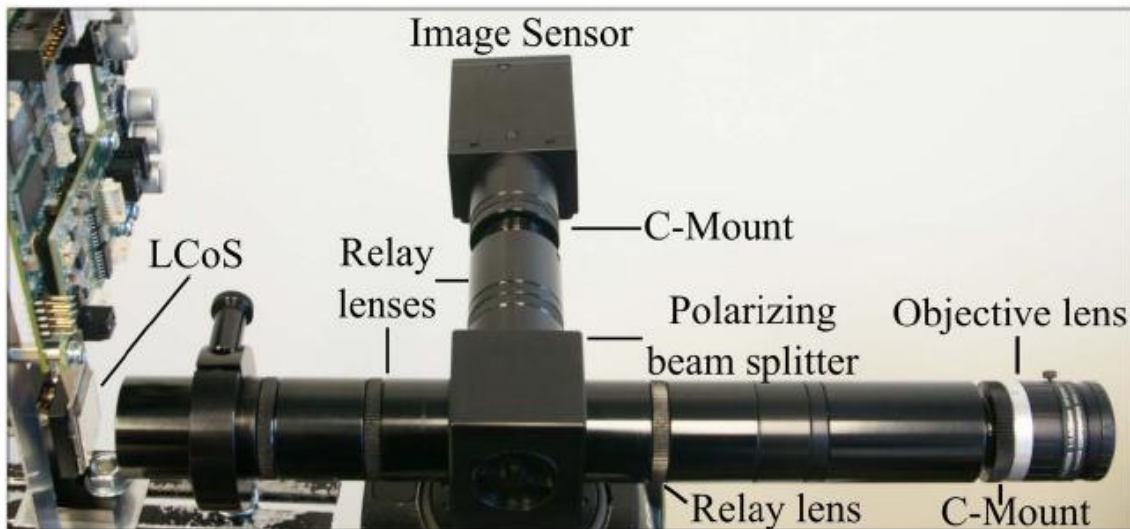


Bilinear interpolation – Uses simple grid-based down-sampling of space-time volume.



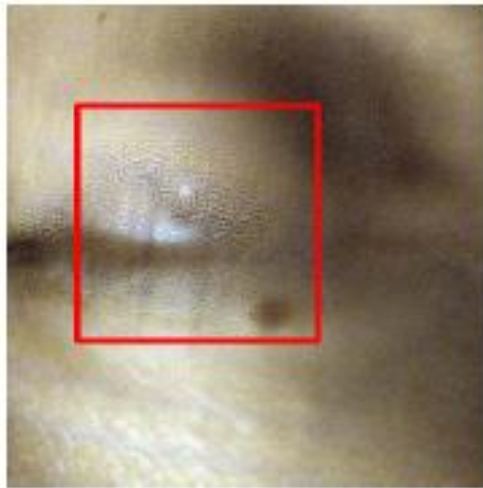
(a) Optical diagram of our setup

Per-pixel binary codes were implemented using a liquid crystal on silicon device (LCoS)



(b) Image of our setup

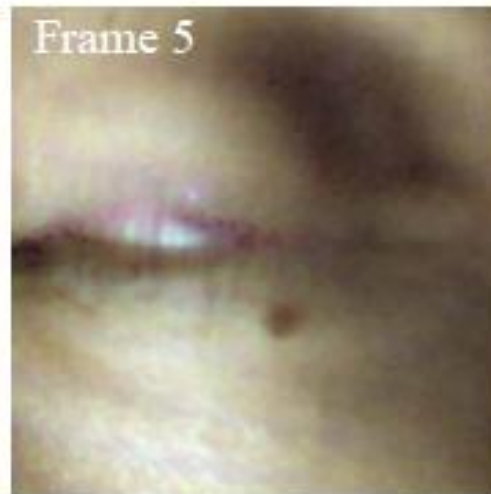
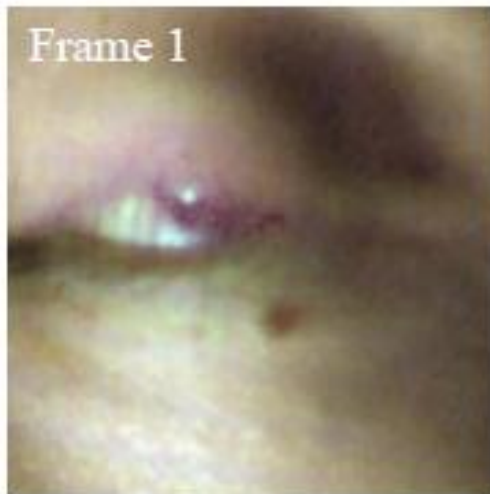
LCoS is synced with the camera and operates at 9-18 times camera frame rate



Coded Exp. Image (27ms)



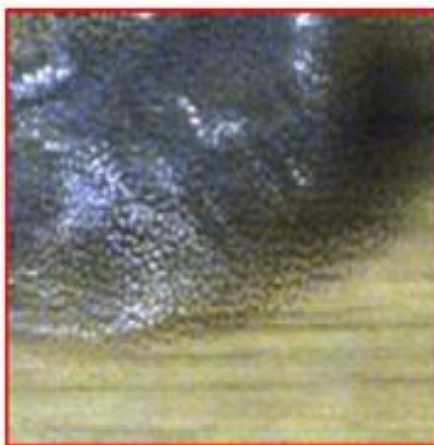
Close-up



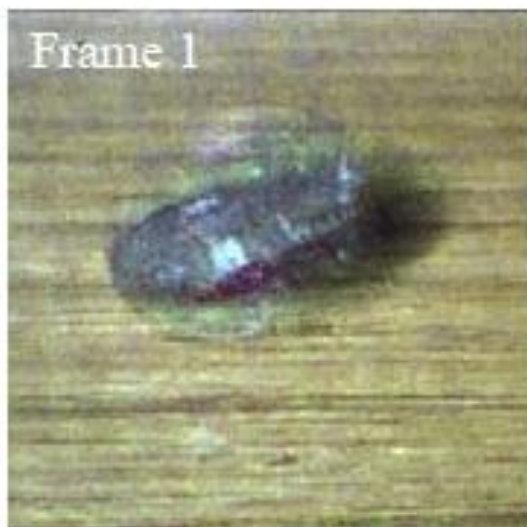
Reconstructed Frames (3 out of 9)



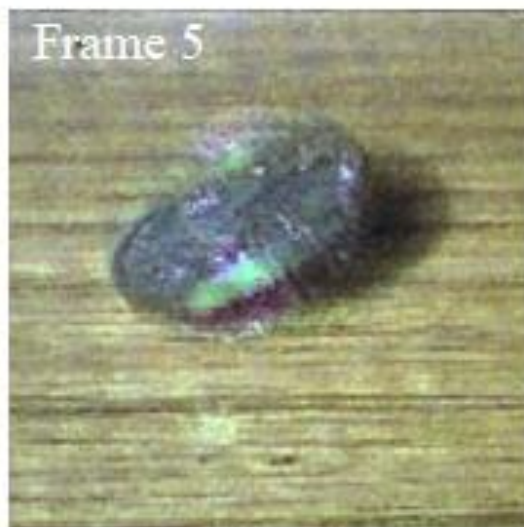
Coded Exp. Image (27ms)



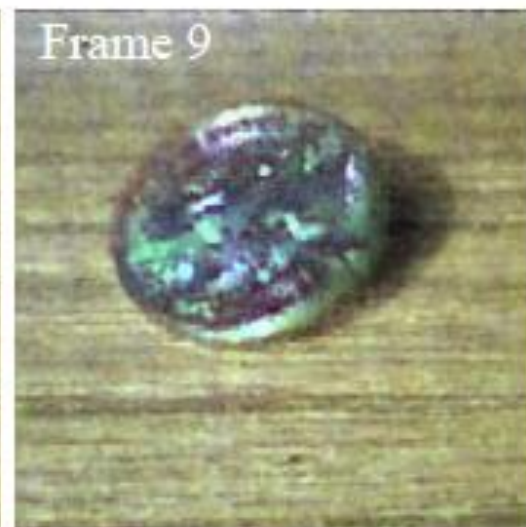
Close-up



Frame 1

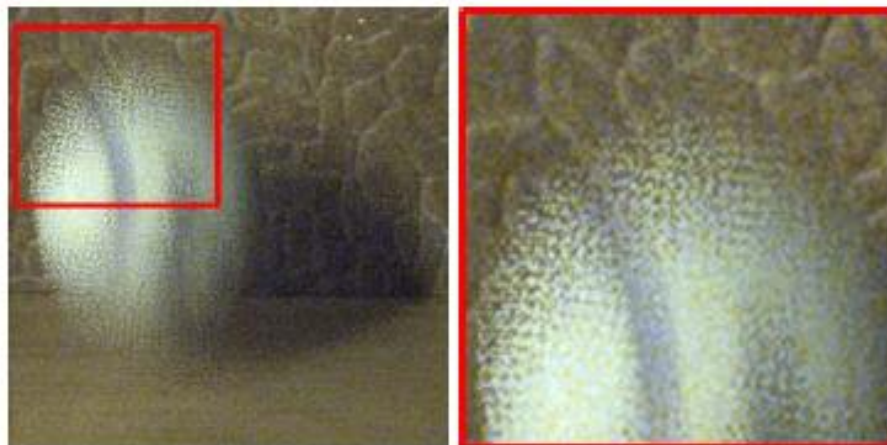


Frame 5



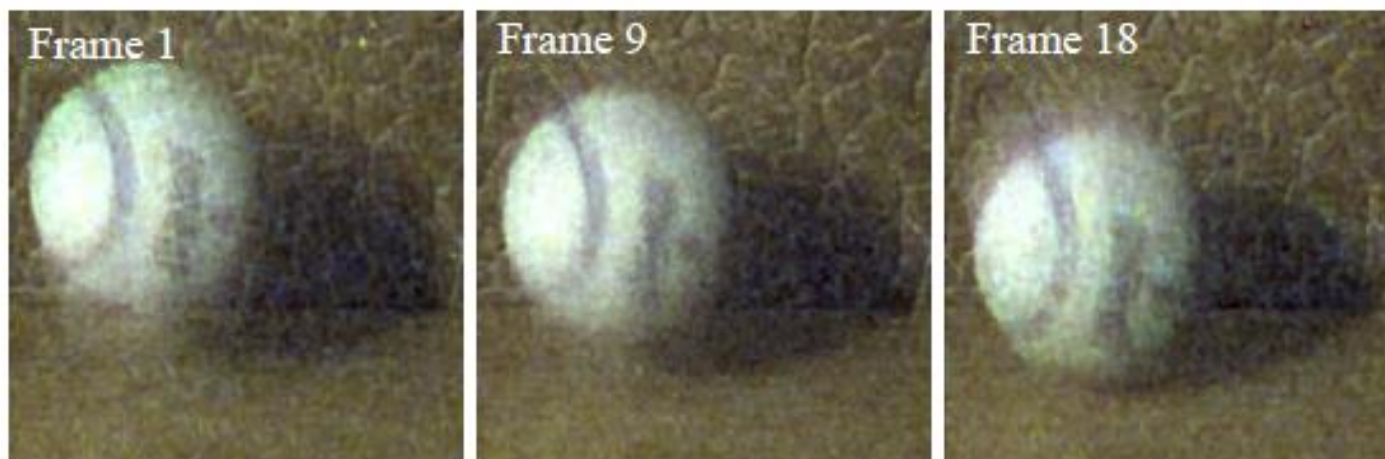
Frame 9

Reconstructed Frames (3 out of 9)



Coded Exp. Image (18ms)

Close-up



Reconstructed Frames (3 out of 18)

Prior Work

- Related hardware prototype in “*P2C2: Programmable Pixel Compressive Camera for High Speed Imaging*”, by Reddy et al, CVPR 2011.
- One major difference – reconstruction technique: sparsity on the transform coefficients of each *sub-frame* + brightness constancy assumption / optical flow for temporal redundancy

Conclusion

- Overcomes space-time tradeoff using per-pixel coded exposure pattern
- Hardware prototype developed
- Works well for varied complex motions (does not require analytical motion model)
- Limitation 1: Maximum target frame-rate must be fixed (e.g. 36X)
- Limitation 2: Requires training videos (which are hopefully `representative') at target frame rate.