INSTITUTE OF TELECOMMUNICATIONS

MASTER THESIS

Focused Single-Pixel Imaging

AUTHOR: *Shahin* GHARAPET *(1427870)*

SUPERVISOR: UNIV.PROF. DIPL.-ING. DR.-ING. NORBERT GÖRTZ

November 15, 2018

ABSTRACT

The Nyquist sampling theorem states that in order to be able to completely recover and reconstruct an analog signal we can use uniform sampling with a rate of twice the bandwidth of the band-limited signal. Miscellaneous types of natural or man made signals exists that have large bandwidth but their information content is relatively small. Following the Nyquist sampling theorem, one has to acquire many samples from the original signal before compressing techniques are deployed to store or transmit the reduced version of it. Compressed sensing (CS) tells us that instead of putting so much effort into sampling the signal with high rate and then discarding a considerable part of it we can specialize the Nyquist theorem to sparse signals and do a considerably smaller amount of sampling and afterwards hope to recover the original signal using sophisticated recovery schemes which is the basic idea of compressed sensing. Among many applications of compressed sensing, this work narrows down its scope to CS-based digital imaging, namely single-pixel imaging. Thanks to single-pixel imaging contrary to conventional cameras, we do not to take a large number of samples from the underlying scene that equals the large number N of pixels on the camera's CCD array. Instead we do $M << N$ linear measurements from the scene and use the image's compressibility property to recover the image.

Recently scientists, in an effort to imitate human vision system, have even moved one step forward and introduced focusing or foveation in images using compressed sensing techniques which is termed as CS-based foveated imaging. The term foveation means to introduce variable spatial resolution to the image. By deploying CS-based fovetaed imaging, one can acquire $M << N$ samples from the scene in a way that after reconstruction some area of higher importance in an image is displayed with higher resolution while other parts are displayed with less resolution. Before emergence of CS-based foveated imaging in digital imaging its primitive mode, namely unfoveated or uniform CS-based digital imaging, was the center of attention. Unfoveated single-pixel imaging simply means that all area of image are recovered with the same resolution and there is no area of higher interest within the image and all parts of the underlying image are treated equally when it comes to the importance or resolution.

This thesis starts with an introduction to single-pixel imaging. The second chapter provides background on state of the art approaches to uniform or unfoveated single-pixel imaging and the underlying sampling and recovery schemes. Chapter 3 represents foveated single-pixel imaging and underlying sampling and recovery techniques. Finally chapter 4 presents an implementation of a typical single-pixel imaging in Matlab which employs several foveated and unfoveated sampling methods as well as underlying recovery techniques. In terms of novelty, it introduces patch-based focusing technique which falls into the category of foveated single-pixel imaging and conducts measurements to evaluate and compare it's performance against it's counterpart, namely superpixel-based imaging technique. In general the thesis seeks to answer three critical questions in the area of the single-pixel imaging based on the numerical analysis and simulations.

Contents

NOTATION

Variables:

- a stands for a random variable.
- a stands for a scalar.
- $\mathbf{a}$ stands for a random vector.
- $\boldsymbol{a}$ stands for a vector.
- $\mathbf{A}$ stands for a random matrix.
- $\boldsymbol{A}$ stands for a matrix.
- $\mathbf{A}_{ij}$ stands for the entry on the $i'th$ row and $j'th$ column of the matrix $\mathbf{A}$.
- $\boldsymbol{A}^T$ stands for transpose of matrix $\boldsymbol{A}$.

1 Introduction

We try to turn analog data from our physical world to digital data by sampling in order to process them. For instance the cameras in our cellphones today sample the light from objects. Nyquist sampling theorem tells us that, if we sample densely enough, namely twice the highest frequency in the analog signal we can recover perfectly the original signal [1]. For instance the digital camera needs to take a number of samples equal to the number of mega pixels on the CCD array which is a huge number! Thanks to the contribution of compressed sensing we can make considerably fewer samples from the image if it is compressible, i.e., it can be represented in a sparse fashion in terms of some basis. In fact compressed sensing enables us to specialize the Nyquist theorem in digital imaging and define a new class of sampling or sensing as sub-Nyquist image acquisition. In fact this is true that in practice many images, specially the ones that do not have many edges, are sparse in terms of a DCT [1] and wavelet basis.

The Figure below depicts a basic digital camera based on single-pixel imaging developed by Rice university.

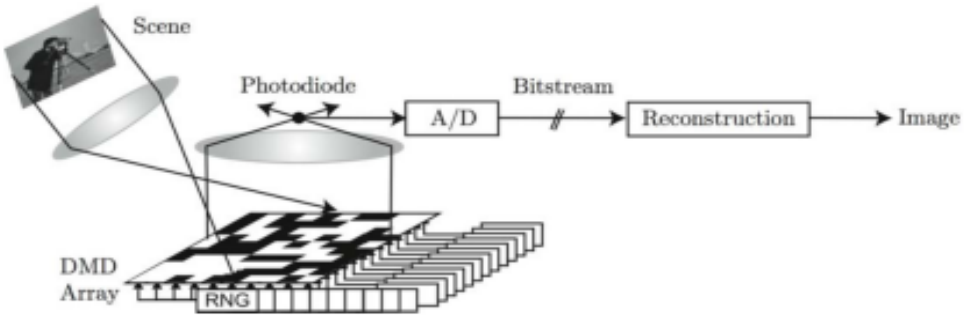


Fig. 1.1. *Rice single-pixel camera [2].*

The proposed design is composed of a digital micromirror device (DMD), lenses and an exotic single photon detector. First, the light from the objects is projected on a DMD array which comprises N tiny mirrors each of them representing a single pixel. Each of these mirrors can be individually oriented to either direct the light to the secondary lens located above the DMD array or away from the secondary lens. As a result some of the light is diverted away from the second lens and some of it is directed towards it. The strength of the resulting light, which is the superposition of the lights directed toward the secondary lens, is sensed by a photodectector and digitized for further processing. This can be interpreted as the inner product of the light from the object and the pattern from the DMD array. The collected light by the second lens is then sensed by a single photo detector and digitized and saved or transmitted to the receiver to be processed and recovered.

After a series of $M << N$ inner products are acquired, digital signal processing can recover the image using compressed sensing recovery techniques. Since the random

[1] Discrete Cosine Transform

pattern on the DMD array is changed for each inner product operation, we have the inner product of the image with a series of independent varying patterns and this helps us to recover the image again. The single-pixel imaging combines both the sampling and compression stages in one step and we are not confronted any more with huge pile of sampling data to process or store.

Among many advantages of single-pixel imaging are:

- ⋆ The measurement and transmission bandwidths are reduced considerably. This means that the sensor can simply sample the source randomly and send it wirelessly to the receiver side which has to do computationally intensive task of image reconstruction. Consequently, this sensing technique is considerably more demanding on the receiver side than the transmitter side and is suitable for applications in which the sensor should not pose high demand on complexity and battery usage which is usually the case.
- ⋆ The size, complexity and cost of resulting camera is reduced.
- ⋆ The quantum efficiency of single photodetector is higher than the pixel sensors on conventional CCD or CMOS sampling arrays.
- ⋆ A single photo detector receives on average more photons in comparison with multi-pixel sensors which leads to higher design immunity against noise.

2 Unfoveated single-pixel imaging

An unfoveated CS-based imaging system is based on uniform compressed sensing and recovery of the images. In other words there is no area of higher resolution or detail in the image:all areas are equally important and recovered with the same resolution. This chapter starts with some basic mathematical tools which are needed to explain the dynamics of CS-based imaging. Then it will go through the CS-based sampling in more detail and finally sheds light on some common CS-based recovery methods.

2.1 Vector spaces and basic notation

Throughout this work we will view our signals as vectors in N dimensional normed euclidean vector space. The definition for the L_p norm is:

$$\|x\|_p = \sqrt[p]{\sum_{n=1}^{\infty} |x|^p} \qquad p = 1,2,...,N \tag{1}$$

$$\|x\|_p = max|x_i| \qquad p = \infty \tag{2}$$

As we already know an indispensable part of any linear vector space is a set Ψ of independent vectors called basis vectors Ψ_i. Let's consider each Ψ_i as column vector of size $N \times 1$. Stacking all these column basis vectors vertically we get the $N \times N$ basis matrix or dictionary Ψ . Any vector x in such a space can be defined as linear combination of the basis vectors, i.e.,

$$\mathbf{x} = \sum_{n=1}^{\infty} s_i * \mathbf{\Psi}_i \quad or \quad \mathbf{x} = \mathbf{\Psi} * \mathbf{s} \tag{3}$$

In the world of digital imaging DCT, DFT and DWT bases are typically used. They are good choices because many images tend to be sparse in terms of these basis functions. In what follows a brief review is given on some of these basis functions:

2.1.1 Discrete Cosine Transform (DCT)

The DCT transforms a signal from a spatial representation into a frequency representation. It uses only real valued cosine functions.

$$\mathsf{s}_k = \alpha_k \sum_{n=0}^{N-1} \mathsf{x}_n cos(\frac{\pi}{N}(n+\frac{1}{2})k) \qquad \text{k = 0, 1, ..., N - 1} \tag{4}$$

$$\mathsf{x}_n = \sum_{0}^{N-1} \alpha_k \mathsf{s}_k cos(\frac{\pi}{N}(n+\frac{1}{2})k) \qquad \text{k = 0, 1, ..., N - 1} \tag{5}$$

$$\alpha_k = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{2}{N}} & k \neq 0 \end{cases} \tag{6}$$

The DCT transform and its inverse in matrix notation:

$$\begin{aligned} \mathbf{s}_N &= \boldsymbol{D}_{[N\times N]} \times \mathbf{x}_{[N]} \\ \mathbf{x}_N &= \boldsymbol{D}^T_{[N\times N]} \times \mathbf{s}_{[N]} \end{aligned} \tag{7}$$

The entries of DCT matrix are calculated as follows:

$$d_{0n} = \frac{1}{\sqrt{N_I}} \quad , \quad d_{kn} = \frac{2}{\sqrt{N_I}} \times \cos(\frac{k(2n+1)\pi}{2N_I}) \quad , \quad k = 1,2,...,N \tag{8}$$

For instance, a 4×4 DCT matrix reads:

$$\begin{bmatrix} d_{00} & d_{01} & d_{02} & d_{03} \\ d_{10} & d_{11} & d_{12} & d_{13} \\ d_{20} & d_{21} & d_{22} & d_{23} \\ d_{30} & d_{31} & d_{32} & d_{33} \end{bmatrix}$$

2.1.2 Discrete Fourier Transform (DFT)

Like the DCT the DFT also maps the signal or image from the time or spacial domain to the frequency domain. In contrast with the DCT in which the signal is represented by a series of harmonically related real-valued basis functions or basis vectors, The DFT exploits a series of harmonically related exponential functions or basis vectors to represent the signal. The DFT and inverse DFT are calculated using:

$$\mathsf{s}_k = \sum_{n=0}^{N-1} \mathsf{x}_n e^{-j2\pi\frac{kl}{N}} \qquad \text{k, l = 0, 1, ..., N - 1} \tag{9}$$

$$\mathsf{x}_n = \frac{1}{N}\sum_{0}^{N-1} \mathsf{s}_k e^{+j2\pi\frac{kl}{N}} \qquad \text{k, l = 0, 1, ..., N-1} \tag{10}$$

2.1.3 Discrete Wavelet Transform (DWT)

DFT and DCT can not represent abrupt changes in signals accurately in general. Because they tend to represent data as a sum of sine waves which oscillate forever and are not localized in time or space. As a result in order to represent the analyze signals that exhibit abrupt changes like images we need to use another class of basis functions that are well localized in time and frequency. A wavelet is a zero mean signal which starts from zero and after some wave like behavior again dies out: as a result it's well localized in time and frequency. In DWT we aim at decomposing the original signal into high and low frequency parts using orthogonal basis of functions which are derived from an original function called the mother wavelet. There are several types of the mother wavelets available. Depending on application and characteristic of the underlying original signal a suitable mother wavelet, which represents the original signal better than other classes of wavelets, is chosen.

JPEG2000 deploys the discrete wavelet transform for image compression [3].

Where the child wavelet $\psi_{j,n}$ is a scaled and shifted version of the original or mother wavelet ψ:

$$\psi_{j,n}(t) = \sqrt{2^{-j}}\psi(2^{-j}t - n), \tag{11}$$

Now we have that $W_0 = span\{\psi(t-n), n \in Z\}$ and the shifted versions have the property $\langle \psi_{(0,k)}, \psi_{(0,l)} \rangle = \delta_{k,l}$. In other words the shifted versions of the scaling function are orthonormal and these basis functions together span the space W_0.

The $\phi_{j,n}$ is a scaled and shifted version of the scaling function ϕ:

$$\phi_{j,n}(t) = \sqrt{2^{-j_0}}\phi(2^{-j_0}t - n), \tag{12}$$

Now we have that $V_0 = span\{\phi_{0,n}, n \in Z\}$ and the shifted versions have the property $\langle \phi_{(0,k)}, \phi_{(0,l)} \rangle = \delta_{k,l}$. In other words the shifted versions of the scaling function are orthonormal and these basis functions together span V_0 space. The scaling function has also the property that:

$$...V_2 \subset V_1 \subset V_0 \subset V_{-1}... \tag{13}$$

Now there is an interesting property that $V_j \cup W_j = V_{j-1}$ In other words the orthogonal complement sets V_j and W_j join together to give birth to the new V_{j-1} space which provides a finer resolution of the signal.

The approximation of the function 'f' at scale l takes the form:

$$f_{-1} = \sum c_n^{(-1)} \phi_{-1,n} = \underbrace{\sum_{n=1}^{\infty} \langle f, \phi_{0,n} \rangle \phi_{0,n}}_{\sum_{n=1}^{\infty} \langle f, \phi_{1,n} \rangle \phi_{1,n} + \sum_{n=1}^{\infty} \langle f, \psi_{1,n} \rangle \psi_{1,n}} + \sum_{n=1}^{\infty} \langle f, \psi_{0,n} \rangle \psi_{0,n} \tag{14}$$

In two dimension, the wavelets are:

$$\psi^d_{j,m,n}(x,y) = \psi_{j,m}(x)\psi_{j,n}(y) \tag{15}$$
$$\psi^v_{j,m,n}(x,y) = \psi_{j,m}(x)\phi_{j,n}(y) \tag{16}$$
$$\psi^h_{j,m,n}(x,y) = \phi_{j,m}(x)\psi_{j,n}(y), \tag{17}$$

Where the scaling function is:

$$\Phi_{j,m,n}(x,y) = \phi_{j,m}(x)\psi_{j,n}(y) \tag{18}$$

The Figure 2.1 demonstrates the Original image and its Daubechies wavelet transform[4].

(a) **(b)**

Fig. 2.1. a) Lena's image. b) Corresponding Daubechies Wavelet coefficients [4].

The Figure 2.1b has some interesting properties. In fact $\{\left|\langle I, \psi^k_{j,m,n} \rangle\right|\}_{j,m,n}$ k $\in \{v,h,d\}$ characterizes the coefficients shown in Fig 2.1b. The buttom right square carries the diagonal coefficients for the matrix $\{\psi^d_{0,m,n}\}_{m,n}$ and shows detail in diagonal direction.

The buttom left square carries the horizontal coefficients for the matrix $\{\psi^h_{0,m,n}\}_{m,n}$ and shows detail in horizontal direction The top right square carries the horizontal coefficients for the matrix $\{\psi^v_{0,m,n}\}_{m,n}$ and shows detain in vertical dimension.

As we move deep into higher resolution space i.e. in V_j, j scale becomes smaller and smaller, our higher scale coefficients are located reside in the left and top most box

which is only function of the ϕ function which resembles the low-pass version of the image and carries most of the energy and can be accepted as a coarse approximation of the whole image.

2.2 Compressive sensing measurement

Normally if we want to follow transform coding (Nyquist rate sampling) we need to get as many as N samples from the original signal x and calculate the coefficients using $s[i] = \langle x, \psi_n \rangle$ where $\{\psi_n\}_{n=1}^{N}$ represents our basis vectors. As we already explained all of these coefficients are quite small except some small number of them. The encoder searches for K largest coefficients and encodes their respective location and magnitude and forgets about the rest of coefficients: that's why it can be regarded as lossy compression. Here the encoder must go through the cumbersome task of computing all N sample values even though it will only keep K of them.

A better approach is to assume that the original signal is k-sparse itself or in terms of some basis and instead of putting so much effort to get N sample points which can be quite a big value, get directly m linear measurements from the signal which brings it directly to compressed form. This simplifies sampling and compression in one step which relaxes signal processing and storage requirements at the sensor or transmitter. In other words the sensing can be described in terms of inner product of our original signal x with a series of M patterns $\{\phi_m\}_{m=1}^{M}$ as in $y[i] = \langle x, \phi_m \rangle$ and stack the result y_m vertically into the vector y. If we stack each of these test vectors horizontally we get the matrix Φ which is also know as measurement or sensing matrix. Of course we add noise vector $\mathbf{w}$ to model the noise in our measurement system. This operation in matrix form is:

$$\mathbf{y} = \mathbf{\Phi}_{[M\times N]} \times \mathbf{x}_{[N]} = \mathbf{\Phi}_{[M\times N]} \times \underbrace{\mathbf{\Psi}_{[N\times N]} \times \mathbf{s}_{[N]}}_{\text{basis expansion of x}} + \mathbf{w}_{[M]} \tag{19}$$

Thus, in compress sensing we multiply our original signal (which lives in the R^N space) with a sensing matrix ,whose number of rows is much smaller than those of the columns, which reduces the number of rows in the sampled signal and reduces the dimension from N to K which is the dimension in which the sample vector $\mathbf{y}$ lives. As a result of this dimensionality reduction also less noise is collected in our measurement which is one of the benefits of CS-based sampling.

In fact this operation serves as a dimensionality reduction. Matrix $\mathbf{\Theta}$, the product of the sensing matrix and the basis or dictionary, is again a fat and short matrix as illustrated in the Figure 2.2.

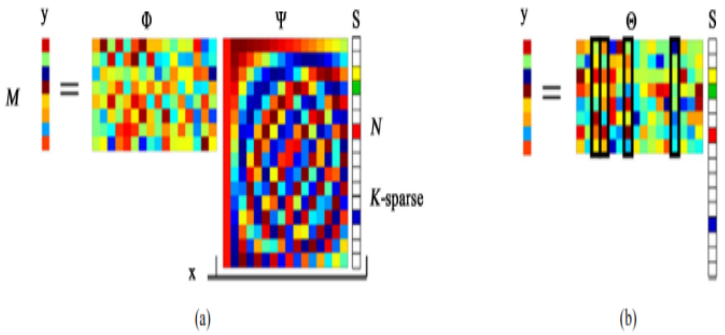


Fig. 2.2. *a) Compressed sensing of a compressible signal or image. b) Resulting sensing matrix* $\mathbf{\Theta}$ *defined as product of the original sensing matrix and the basis matrix [2].*

Since the number of rows are fewer than the number of columns we are confronted with an underdetermined system of linear equations which means that it can have infinitely many solutions. Apparently we are looking for a unique solution. For instance we do not want the compressed sensing of two images be mapped to the same sampling vector $\mathbf{y}$. If this happens, how can we hope that we recover the result back to two original images? The key here is that the vector $\mathbf{s}$ is k-sparse, as a result this original fat and short matrix turns into a skinny and tall matrix which means we are not restricted with the underdetermined system anymore and we can exactly recover the original k-sparse signal. In other words if we have two k sparse vectors $\mathbf{s}^{'}$ and $\mathbf{s}^{''}$:

$$\begin{aligned} &\mathbf{\Phi} \times \mathbf{s}^{'} \neq \mathbf{\Phi} \times \mathbf{s}^{''} \\ &\mathbf{\Phi} \times \left(\mathbf{s}^{'} - \mathbf{s}^{''}\right) \neq 0 \end{aligned} \tag{20}$$

The difference of any two sparse vectors must not lie in null space of matrix $\mathbf{\Phi}$ and the distance of sparse vectors are preserved through the transfer using matrix $\mathbf{\Phi}$. Hence if we pick randomly any $2k$ of vectors for matrix $\mathbf{\Phi}$ they must be independent. This means the column space of the matrix must be at least of dimension $2k$ or twice the sparsity level. Consequently, we should be careful while designing the compressed sensing matrix and not any kind of matrix can be used in this respect. In general, the sensing matrix should satisfy the Restricted Isometry Property (RIP) of order $2k$. The number of linear measurements M from the original signal which is equal to the number of rows of the measurement matrix then is calculated as [5]:

$$M \geq \frac{2k}{10} \times \log(\frac{N}{k}) \tag{21}$$

As is evident from the Figure 2.2b, the final sensing matrix is the product of two matrices ϕ and *Psi*. As far as the matrix Φ is i.i.d Gaussian and the columns of the matrix ψ or our basis vectors are orthogonal, the resulting sensing matrix $\Phi \times \Psi$ is again i.i.d Gaussian still satisfying the RIP criteria.

2.3 Images and DCT basis expansion

Now let's consider the analysis of an image as a two dimensional signal in the DCT domain. As I already discussed, for many images (like natural images) we have a lot of smooth areas and few edges and consequently can be really sparsely represented in terms of the DCT basis. The JPEG standard for instance makes use of a DCT basis to do image compression. It does so by expanding the image in terms of the DCT basis and keeping those DCT coefficients with large magnitudes and discarding all others. This can bring about a compression rate of 3%.

The Figure 2.3 depicts the typical DCT coefficients for an image.

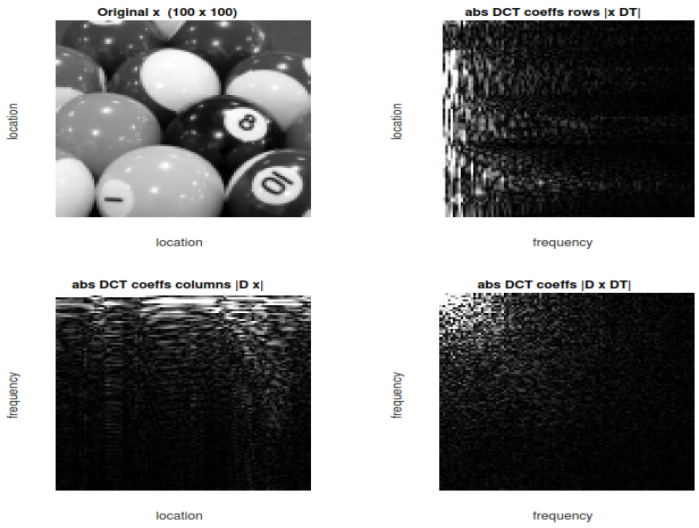


Fig. 2.3. *Top left: Original image. Top right. Bottom left: DCT coefficients in space and frequency domains. Bottom right: DCT coefficients in the frequency domain [6].*

As one can observe from the Figure 2.3, at the bottom right, the majority of the DCT coefficients is zero except some in upper left hand corner of the DCT coefficient matrix. This as well indicates that much of energy for the image is concentrated in low frequencies.

Now let's assume that a 4×4 matrix $\mathbf{X}$ represents our image in gray scale format:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

First we need to form the $N \times N$ DCT matrix $\boldsymbol{D}$:

$$\boldsymbol{D} = \begin{bmatrix} d_{00} & d_{01} & d_{02} & d_{03} \\ d_{10} & d_{11} & d_{12} & d_{13} \\ d_{20} & d_{21} & d_{22} & d_{23} \\ d_{30} & d_{31} & d_{32} & d_{33} \end{bmatrix}$$

whose entries are calculated as follows:

$$d_{0n} = \frac{1}{\sqrt{N_I}} n \qquad = 0,1,2,...,N \tag{22}$$

$$d_{kn} = \frac{2}{\sqrt{N_I}} \times \cos(\frac{k(2n+1)\pi}{2N_I}) n \qquad = 0,1,2,...,N \tag{23}$$

Then the DCT coefficients are calculated as:

$$\mathbf{S} = D \times \mathbf{X} \times D^T \tag{24}$$

The inverse DCT transform reads:

$$\mathbf{X} = D^T \times \mathbf{S} \times D \tag{25}$$

Alternatively the vectorized version of the coefficient matrix can be calculated as:

$$vec(\mathbf{S}) = D_{kron} \times vec(\mathbf{X}) \tag{26}$$

$$D_{kron} = D \otimes D \tag{27}$$

$$vec(\mathbf{S}) = \begin{bmatrix} s_{00} \\ s_{10} \\ s_{20} \\ s_{30} \\ s_{01} \\ s_{11} \\ s_{21} \\ s_{31} \\ s_{02} \\ s_{12} \\ s_{22} \\ s_{32} \\ s_{03} \\ s_{13} \\ s_{23} \\ s_{33} \end{bmatrix} = \underbrace{\begin{bmatrix} d_{00}D & d_{01}D & d_{02}D & d_{03}D \\ d_{10}D & d_{11}D & d_{12}D & d_{13}D \\ d_{20}D & d_{21}D & d_{22}D & d_{23}D \\ d_{30}D & d_{31}D & d_{32}D & d_{33}D \end{bmatrix}}_{D_{kron}} \begin{bmatrix} x_{00} \\ x_{10} \\ x_{20} \\ x_{30} \\ x_{01} \\ x_{11} \\ x_{21} \\ x_{31} \\ x_{02} \\ x_{12} \\ x_{22} \\ x_{32} \\ x_{03} \\ x_{13} \\ x_{23} \\ x_{33} \end{bmatrix} \tag{28}$$

As can be seen from next sections, we use the vectorized version of the DCT coefficient matrix $\mathbf{C}$ to be applied to the CS-based sensing or measurement system.

2.4 Properties of the sensing matrix

Sensing matrices generally fall into two categories:

- ⋆ Random matrices:

  These matrices are generated using identical and independent distributions (i.i.d) such as Gaussian and Bernoulli. Generally these are easy to construct and the probability of reconstruction is high when these matrices are used as sensing matrix to sample from the original signal. However they need a lot of storage space and the recovery problem becomes difficult when the original signal or image is considerably large. It's worthwhile to mention that because of their random nature no specific algorithm is there to speed up matrix multiplication.

- ⋆ Deterministic matrices:

  Usually the signal reconstruction becomes less complex when deterministic matrices are used and also less storage space is needed.

2.5 Compressed sensing recovery

Now let's recall our compressive sensing model:

$$y = \mathbf{\Phi}_{[M\times N]} \times \mathbf{x}_{[N]} = \mathbf{\Phi}_{[M\times N]} \times \underbrace{\mathbf{\Psi}_{[N\times N]} \times \mathbf{s}_{[N]}}_{\text{basis expansion of x}} + \mathbf{w}_{[M]} \tag{29}$$

Where $\mathbf{\Theta}$ is the product of the measurement or sensing matrix $\mathbf{\Phi}$ with sampling ratio $R = \frac{M}{N}$ and our sparsifying basis $\mathbf{\Psi}$. $\mathbf{x}$ is the sparse or compressible signal that we would like to reconstruct, y is the compressed sensing observation vector and white Gaussian vector with i.i.d entries $w_i \sim \mathcal{N}(0, \sigma_w^2)$.

In contrast with traditional sampling methods, where the linear construction formulas can be used to recover the original signal, in compressive sensing the task of signal recovery is highly nonlinear and more complex in general. In this section several classes of signal recovery are studied.

2.5.1 L1 minimization

Historically the first recovery method is L_1 minimization also known as Basis Pursuit (BP). It has the best sparsity-undersampling trade-off.

If we forget about the noise we need to solve following optimization problem:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{argmin} \|\mathbf{s}\|_0 \quad s.t. \quad \mathbf{y} = \underbrace{\mathbf{\Phi} \times \mathbf{\Psi}}_{\mathbf{\Theta}} \times \mathbf{s} \tag{30}$$

In the presence of the noise this formula becomes:

$$\hat{\mathbf{s}} = \underset{s}{argmin} \|\mathbf{s}\|_0 \quad s.t. \quad \|\mathbf{y} - \mathbf{\Theta s}\|_2 \leq \varepsilon \tag{31}$$

Unfortunately the L_0 minimization problems above is NP hard and there is no known algorithm to solve this problem. However it has been shown [5] that under some constraints problem 30 can be relaxed a bit and reformulated as:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{argmin} \|\mathbf{s}\|_1 \quad s.t. \quad \mathbf{y} = \mathbf{\Theta} \times \mathbf{s} \tag{32}$$

If we want to take into account the effect of additive noise then the formula becomes:

$$\hat{\mathbf{s}} = \underset{s}{argmin} \|\mathbf{s}\|_1 \quad s.t. \quad \|\mathbf{y} - \mathbf{\Theta s}\| \leq \varepsilon \tag{33}$$

This is a convex optimization problem for which interesting algorithms and linear programming solvers (LP) exist [7].

As is evident from above formula, the L_0 norm has been replaced by L_1 norm and still the two problems have the same unique solution provided that the measurement matrix satisfies some criteria and also the original signal is sparse.

The Figure 2.4 demonstrates the penalization that L_0 and L_1 norms pose.

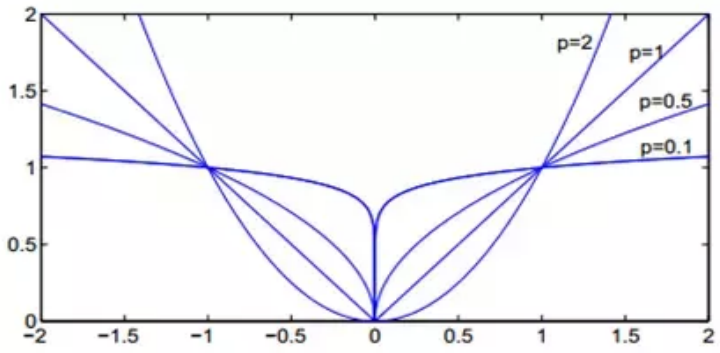


Fig. 2.4. *L_p norm curves. As p tends to zero, the norm approaches constant value one for non-zero values except at the center which is zero [8].*

As we can see as p approaches zero the L_p norm starts to pose no penalty for entries equal to zero. In contrast it penalizes other non-zero entries with a constant value irrespective of their sign and magnitude. L_1 norm treats the penalization linearly. It poses no penalty for the zero valued entries of the vector. But in contrast with the L_0 norm it penalizes non-zero values linearly dependent on the value of the entry. Apparently L_0 and L_1 norms treat the vectors with a minor difference. But the question is how the relaxed version of the recovery problem also leads to the same solution as the original problem. Under some circumstances (level of sparsity and the properties of measurement matrix ($\mathbf{\Theta}$) these two solutions are exactly the same [9]. The relation between the equations has attracted a lot of researches and papers [10, 11, 12].

One drawback for L_1 optimization methods is that when it comes to large scale applications they become inefficient and expensive.

2.5.2 Least Absolute Shrinkage and Selection Operator (LASSO) using the ADMM method

The Alternating Direction Method of Multipliers (ADMM) is actually a method which breaks the main convex optimization problem into smaller and easier to solve optimization problems [13, 14].

In machine learning and machine vision we are frequently confronted with convex or L_1 optimization of the form:

$$\hat{s} = \underset{\mathbf{s}}{argmin}(\frac{1}{2}\|\mathbf{y} - \mathbf{\Theta} \times \mathbf{s}\|_2^2 + \gamma\|\mathbf{s}\|_1) \tag{34}$$

We can simply add a dummy variable $\mathbf{p}$ into the formula and rewrite it as

$$\hat{\mathbf{s}} = \underset{\mathbf{s},\mathbf{p}}{argmin}\|\mathbf{y} - \mathbf{\Theta} \times \mathbf{s}\|_2^2 + \gamma\|\mathbf{p}\|_1 + \|\mathbf{s} - \mathbf{p}\|_2^2 \quad s.t. \quad \mathbf{s} = \mathbf{p} \tag{35}$$

The augmented Lagrangian of the above statement is:

$$L(\mathbf{s},\mathbf{p},\mathbf{r}) = \|\mathbf{y} - \mathbf{\Theta} \times \mathbf{s}\|_2^2 + \gamma\|\mathbf{p}\|_1 + \frac{\rho}{2}\|\mathbf{s} - \mathbf{p}\|_2^2 + vec(\boldsymbol{R})^T vec(\mathbf{s} - \mathbf{p}) \tag{36}$$

The algorithm has three main steps:

At first it minimizes the Lagrangian w.r.t $\mathbf{s}$ as a result the term $\|\mathbf{p}\|_1$ is seen as constant and consequently the problem turns into a simple least squares regression problem. At the second iteration we aim at minimizing w.r.t $\mathbf{p}$ then term $\|\mathbf{y} - \mathbf{\Theta s}\|_2^2$ has no effect and the minimization can be done easily. The final step updates the Lagrangian multiplier matrix $\boldsymbol{R}$.

$$\mathbf{s}^{(t+1)} = \underset{\mathbf{s}}{argmin} \quad L(\mathbf{s}, \mathbf{p}^t, \boldsymbol{R}^t) \tag{37}$$

$$\mathbf{p}^{(t+1)} = \underset{\mathbf{p}}{argmin} \quad L(\mathbf{s}^{(t+1)}, \mathbf{p}, \boldsymbol{R}^t) \tag{38}$$

$$\boldsymbol{R}^{(t+1)} = \boldsymbol{R}^t + \rho(\mathbf{s} - \mathbf{p}) \tag{39}$$

2.5.3 Greedy methods

Greedy methods do not use the L_1 relaxation method and try to solve directly the formula below:

$$\hat{\mathbf{s}} = \underset{s}{argmin}\|\mathbf{s}\|_0 \quad s.t. \quad \|\mathbf{y} - \mathbf{\Theta s}\|_2 \leq \varepsilon \tag{40}$$

Unlike the L_1 minimization method, greedy methods are less computationally complex and therefore considerably faster especially for large scale applications and some variants of the greedy method provide accuracy better than L_1 minimization.

Among prominent greedy approaches are: Matching pursuit (MP) [15], orthogonal matching pursuit (OMP) [16], stagewise OMP (StOMP) [17] and compressive sampling matching pursuit (CoSaMP) [18].

The most basic approach is the matching pursuit (MP). The base idea in the matching pursuit is to find first the most important column vector or atom in the sensing matrix Θ or dictionary . We do so by traveling through the dictionary and choose the most important one that helps to minimize the least square error $\|\mathbf{y}-\mathbf{\Theta s}\|_2$. This translates into finding the atom which is most parallel to the residual $\mathbf{y}-\mathbf{\Theta}\times\mathbf{s}$ or in other words the atom whose inner product with the residual produces larger absolute value. Now as the name greedy implies we keep this selected atom and check whether the resulting least square error is below the threshold ε or not. If not we travel again through dictionary and find the second most important atom that with help of first atom minimizes the least square error. If the error is below some threshold then we quit but if not we keep the first two most important atoms and travel again through other columns of dictionary to choose the third atom and so on.

The procedure for MP algorithm is as follows:

Initialization phase:

$\hat{\mathbf{s}}^{(0)} = \mathbf{0}_{[N\times 1]}$

$\mathbf{r}^{(0)} = \mathbf{y}$

$I^{(0)} = \emptyset$

Normal iterations:

$i_t^* \in \underset{i}{argmax}\left|\langle \mathbf{r}^{(t-1)}, \mathbf{\Theta}_i\rangle\right|$

$I^{(t)} = I^{(t-1)} \cup \{i_t^*\}$

$\hat{\mathbf{s}}^t = \langle \mathbf{r}^t, i_t^*\rangle$

$\mathbf{r}^t = \mathbf{y} - \mathbf{\Theta}\times\hat{\mathbf{s}}^t$

iterations continue until: $\left\|\hat{\mathbf{s}}^{(t+1)}-\hat{\mathbf{s}}^{(t)}\right\|_2 < \varepsilon\left\|\hat{\mathbf{s}}^{(t)}\right\|_2$ with $\varepsilon < 10^{-4}$ (trade-off iterations vs. accuracy).

As is evident from the algorithm above, the initial estimate of the signal is set to zero and the initial residual or error is set equal to $\mathbf{y}$. As the iterations evolve, more and more atoms are added to the active set I.

Orthogonal matching pursuit (OMP) like its predecessor involves many inner products and tries to minimize the residual with help of most important atoms. But it improves the MP algorithm by updating the coefficients extracted so far by projecting the signal $\mathbf{y}$ into the subspace of all already chosen atoms at each iteration which produces indeed more accurate results i.e. sparsest vectors $\mathbf{s}$ which meets minimum least square criteria.

The procedure for OMP algorithm is as follows [19]:

Initialization phase:

$\hat{\mathbf{s}}^{(0)} = \mathbf{0}_{[N\times 1]}$

$\mathbf{r}^{(0)} = \mathbf{y}$

$I^{(0)} = \emptyset$

Normal iterations:

$i_t^* \in \underset{i}{argmax}\left|\langle r^{t-1}, \mathbf{\Theta}_i\rangle\right|$

$I^t = I^{t-1} \cup \{i_t^*\}$

$\hat{\mathbf{s}}^t = (\mathbf{\Theta}_{I_t}^* \times \mathbf{\Theta}_{I_t})^{-1} \times \mathbf{\Theta}_{I_t}^* \times \mathbf{y}$

$\mathbf{r}^t = \mathbf{y} - \mathbf{\Theta} \times \hat{\mathbf{s}}^t$

iterations continue until: $\left\|\hat{\mathbf{s}}^{(t+1)} - \hat{\mathbf{s}}^{(t)}\right\|_2 < \varepsilon \left\|\hat{\mathbf{s}}^{(t)}\right\|_2$ with $\varepsilon < 10^{-4}$ (trade-off iterations vs. accuracy).

As one can observe, the OMP algorithm projects the observation vector $\mathbf{y}$ on the subspace of collected atoms to get the new approximation of the sparse vector $\mathbf{s}$. This is in fact done by calculating the left pseudo inverse of the matrix $\mathbf{\Theta}$ and multiply it from the right side by vector $\mathbf{y}$. Despite OMP operation involves matrix inversion to compute the pseudo norm, it's still faster than L_1 minimization method.

2.5.4 Iterative thresholding

As we already understood convex optimization problem becomes inefficient and expensive in large scale applications. As an alternative iterative thresholding can be used to improve the speed as it's easy and requires less storage. However this technique exhibits a worse sparsity-undersampling trade-off than convex optimization. The steps in iterative thresholding read:

$$\begin{aligned} \mathbf{s}^{(t+1)} &= \eta(\mathbf{s}^{(t)} + \boldsymbol{\theta}^T \mathbf{z}^{(t)}; \lambda^{(t)}), \\ \mathbf{z^{(t)}} &= \mathbf{y} - \mathbf{\Theta}^T \mathbf{s}^{(t)} \end{aligned} \tag{41}$$

Here $s^{(t)}$ is current estimate of the signal and $\mathbf{z}^{(t)}$ is current estimate of residual. $\langle \boldsymbol{u} \rangle = \sum_1^N \frac{\boldsymbol{u}(i)}{N}$ where $\boldsymbol{u} = (\boldsymbol{u}(1)\boldsymbol{u}(2)...\boldsymbol{u}(N))$.

To have a closer look:

Initialization phase:

$\hat{\mathbf{s}}^{(0)} = \mathbf{0}_{[N \times 1]}$

$\mathbf{z}^{(0)} = \mathbf{y}$

Normal iterations:

$\mathbf{u}^{(t-1)} = \hat{\mathbf{s}}^{(t-1)} + \mathbf{\Theta}^T \mathbf{z}^{(t-1)}$

$\hat{\mathbf{s}}^{(t)} = \eta(\mathbf{u}^{(t-1)}; \tau)$

$\mathbf{z}^{(t)} = \mathbf{y} - \mathbf{\Theta}\hat{\mathbf{s}}^{(t)}$

iterations continue until: $\left\|\hat{\mathbf{s}}^{(t+1)} - \hat{\mathbf{s}}^{(t)}\right\|_2 < \varepsilon \left\|\hat{\mathbf{s}}^{(t)}\right\|_2$ with $\varepsilon < 10^{-4}$ (trade-off iterations vs. accuracy).

At first step it calculates the new approximation for the solution. At second step it calculates the current residual or error in a ping pong manner. The value for τ which is used for thresholding, needs to be chosen by trial and error which is an disadvantage of this method. We will later see that more sophisticated algorithms use adaptive schemes to find optimal thresholding values.

The function η can be any function. Actually this function plays a big role in determining the convergence of solution. It is a suitable denoiser to estimate the sparse signal in presence of the additive Gaussian noise. If no function is used the solution converges to least L_2 norm and not the least L_1 norm which is not acceptable in our compressed sensing framework! The use of soft thresholding induces L_1 norm minimization.

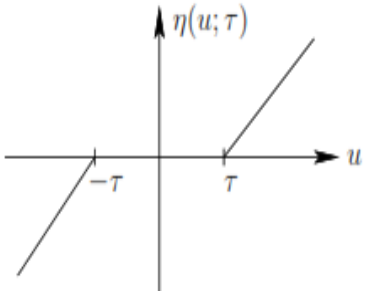


Fig. 2.5. *Soft thresholding function [20].*

$$\eta(u;\tau) = \begin{cases} u-\tau & u > \tau \\ 0 & -\tau \leq u \leq \tau \\ u+\tau & u < -\tau \end{cases}$$

In this case the algorithm is called Iterative Soft Thresholding (IST) [21]. If hard thresholding function is chosen for η as:

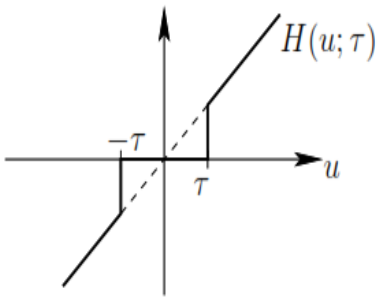


Fig. 2.6. *Hard thresholding function [20].*

$$H(u;\tau) = \begin{cases} u & u > \tau \\ 0 & -\tau \leq u \leq \tau \\ u & u < -\tau \end{cases}$$

Then the algorithm is called Iterative Hard Thresholding (IHT) [22].

2.5.5 Approximate Message Passing (AMP)

In [23, 24, 25, 26] it is proposed that if the residual calculation step in the above mentioned iterative thresholding is modified by adding a term called Onsager reaction term to the residual calculation step then we get the AMP algorithm. This idea comes from the belief propagation in graphical models and introduced byDonoho and co-authors in [25]. This way we have the benefit of increased speed of the iterative thresholding and at the same time the improved sparsity-undersampling trade-off. Extensive numerical and Monte Carlo simulations show that in fact AMP matches those results of the convex optimization or linear programming (LP) reconstruction techniques [19].

The original paper has used the soft thresholding function for function η which acts as a scalar denoiser for the matched filter output $\mathbf{u}^{(i-1)}$. It enforces a sparse solution and assumes that the signal is sparse but no specific knowledge about the probability distribution of the source is available. If the signal prior is also available we use the Bayesian Approximate Message Passing (BAMP) algorithm which delivers more accurate result.

The AMP algorithm reads [25]:

$$\mathbf{s}^{(t+1)} = \eta(\mathbf{s}^{(t)} + \boldsymbol{\theta}^T \mathbf{z}^{(t)}; \lambda^{(t)}), \tag{42}$$

$$\mathbf{z^{(t)}} = \mathbf{y} - \boldsymbol{\Theta}^T \mathbf{s}^{(t)} + \underbrace{\frac{1}{\gamma} \mathbf{z}^{(t-1)} < \eta'(\boldsymbol{\Theta}^T \mathbf{z}^{(t-1)} + \mathbf{s}^{(t-1)}) >}_{\text{Onsager term}} \tag{43}$$

To have a closer look into algorithm [6]:

initialization phase:

$\hat{\mathbf{s}}^{(0)} = 0_{[N\times 1]}$ Signal vector

$\mathbf{z}^{(0)} = \mathbf{y}$

$c^{(0)} = \frac{1}{M}\|\mathbf{z}^{(0)}\|_2^2$

normal iterations:
For i = 1,2,3, ...

$\mathbf{u}^{(t-1)} = \hat{\mathbf{s}}^{(t-1)} + \boldsymbol{\Theta}^T \mathbf{z}^{(t-1)}$ Substitute measurements

$\hat{\mathbf{s}}^{(t)} = \eta(\mathbf{u}^{(t-1)}; \sqrt{\beta c^{(t-1)}})$

$\mathbf{z}^{(t)} = \mathbf{y} - \boldsymbol{\Theta}\hat{\mathbf{s}}^{(t)} + \mathbf{z}^{(t-1)} \frac{1}{M} \left\| \hat{\mathbf{s}}^{(t)} \right\|_0$ Residual computation

$\hat{\mathbf{c}}^{(t)} = \frac{1}{M}\|\mathbf{z}^{(t)}\|_2^2$

iterations continue until: $\left\| \hat{\mathbf{s}}^{(t+1)} - \hat{\mathbf{s}}^{(t)} \right\|_2 < \varepsilon \left\| \hat{\mathbf{s}}^{(t)} \right\|_2$ with $\varepsilon < 10^{-4}$ (trade-off iterations vs. accuracy).

Intuitively, what the AMP algorithm does is to successively denoise the output of $\hat{\mathbf{s}}^{(t-1)} + \boldsymbol{\Theta}^T \mathbf{z}^{(i-1)}$ at each iteration step and tries to minimize the MSE at each iteration step until it falls below a threshold when algorithm finishes. In other words, we observe a white noise (w_j) corrupted signal $u_j^{(i-1)} = s^{(j-1)} + \boldsymbol{\Theta}^T z^{(i-1)}$ at each iteration that the

noise variance diminishes little by little as the iterations go on [19, 25, 27]. In fact the residual vector $\mathbf{z}^{(t)}$ can be well modeled as an i.i.d additive white gaussian vector [23]. It's well understood that the residual of the AMP shows a Gaussian distribution. This is not the case with the Iterative Soft Thresholding (IST) discussed earlier. In many situations in compressed sensing we are not aware of the probability distribution of the source under sample. Then we have to rely on the AMP which uses soft thresholding as a general guideline for all signals that are sparse in some base. If we already have the knowledge of the signal prior, then we rely on BAMP method which is described next. One disadvantage of AMP is that the value for β needs to be found by trial and error which makes it less efficient than BAMP. It should be noted that the value for β needs to be chosen by trial and error which is a disadvantage of this method.

2.5.6 Bayesian Approximate Message Passing (BAMP)

Provided that the prior distribution of the original signal is known, we can use this knowledge to specialize the AMP algorithm to BAMP and take advantage of MMSE denoiser which can adapt itself to the statistics of each of the entries of the signal vector individually. This technique was first proposed in [28] which achieves more accuracy in the solution in comparison with its AMP counterpart.

As is evident from the Figure 2.7, there is a link between the frequency content of the image and its corresponding DCT coefficient matrix. BAMP method can exploit this relation to extract distribution of the entries in DCT coefficient matrix. This leads to a more accurate solution in comparison with AMP which ignores the signal prior.

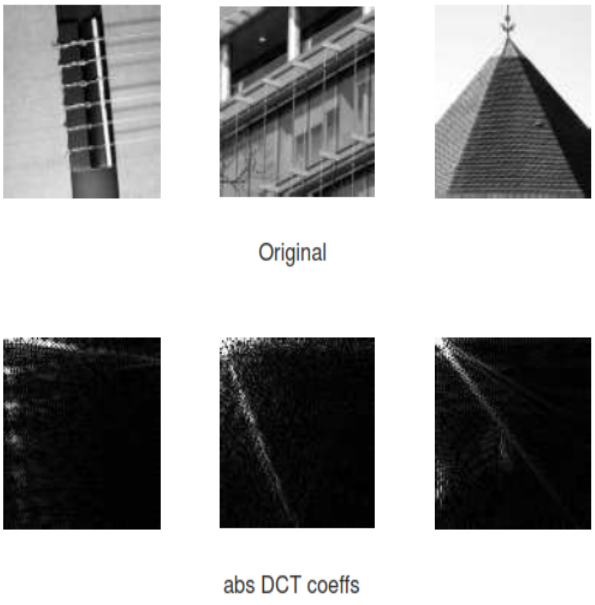


Fig. 2.7. *Different patterns and their frequency in images yield specific distribution in their corresponding DCT coefficient matrix [6].*

The algorithm for BAMP is again like AMP with a difference that the denoiser function F is no more universally chosen as soft thresholding function and is dependent on signal prior. The BAMP algorithm proceeds in iterative fashion according to [25]:

$$\begin{aligned}\mathbf{s}^{(t+1)} &= F(\mathbf{s}^{(t)} + \boldsymbol{\theta}^T \mathbf{z}^{(t)}; \lambda^{(t)}),\\ \mathbf{z}^{(t)} &= \mathbf{y} - \boldsymbol{\Theta}^T \mathbf{s}^{(t)} + \underbrace{\frac{1}{\gamma}\mathbf{z}^{(t-1)} < F'(\boldsymbol{\Theta}^T \mathbf{z}^{(t-1)} + \mathbf{s}^{(t-1)}) >}_{\text{Onsager term}}\end{aligned} \tag{44}$$

Let's have a closer look to the algorithm [6]:

Initialization phase:

$\hat{\mathbf{s}}^{(0)} = \boldsymbol{0}_{[N\times 1]}$

$\mathbf{z}^{(0)} = \mathbf{y}$

$\mathbf{c}^{(0)} = \frac{1}{M}\|\mathbf{z}^{(0)}\|_2^2$

Normal iterations:

$\mathbf{u}^{(t-1)} = \hat{\mathbf{s}}^{(t-1)} + \boldsymbol{\Theta}^T \mathbf{z}^{(t-1)}$

$\hat{\mathbf{s}}^{(t)} = F(\mathbf{u}^{(t-1)}; c^{(t-1)})$

$z^{(t)} = \mathbf{y} - \boldsymbol{\Theta}\hat{\mathbf{s}}^{(t)} + \mathbf{z}^{(t-1)} \frac{1}{M}\sum_{j=1}^{N} F'(u_j^{(i-1)}; c^{(t-1)})$

$\hat{c}^{(t)} = \frac{1}{M}\|\mathbf{z}^{(t)}\|_2^2$

iterations continue until: $\left\|\hat{\mathbf{s}}^{(t+1)} - \hat{\mathbf{s}}^{(t)}\right\|_2 < \varepsilon \left\|\hat{\mathbf{s}}^{(t)}\right\|_2$ with $\varepsilon < 10^{-4}$ (trade-off iterations vs. accuracy).

It's worthwhile to mention that intuitively BAMP at each iteration calculates the white Gaussian noise contaminated version of the signal which will be used in next step to calculate the new value for our solution. So it's a signal plus noise model. As a result of the iterations the variance of the noise (noise power) decreases below a threshold value when the iteration ends and the final value for the signal is the solution to the problem. Here we are not dependent on the sparsity of the signal.

The denoising function F is calculated using following formula [6]:

$$F_j(u_j; c) = E_{x_j}(X_j | U_j = u_j) \tag{45}$$

$$F_j'(u_j; c) = \frac{d}{du_j} F(u_j; c) \tag{46}$$

$$F_j'(u_j; c) = \frac{d}{du_j} F(u_j; c) \tag{47}$$

$$p_{S_j|U_j}(s_j|u_j; c) = \frac{p_{S_j|U_j}(s_j|u_j; c)}{p_{U_j}(u_j; c)} = \frac{p_{U_j|S_j}(u_j|s_j; c) p_{S_j}(s_j)}{p_{U_j}(u_j; c)} \quad j = 1, 2, ..., N \tag{48}$$

Here is what differentiate the BAMP from the AMP. In fact

$$p_{U_j|S_j}(u_j|s_j;c) = \frac{1}{\sqrt{2\pi c}} \exp\left(\frac{-1}{2c}(s_j - u_j)^2\right) \tag{49}$$

Let's consider the typical trend in the DCT coefficients of an image which can be observed in the Figure 2.8:

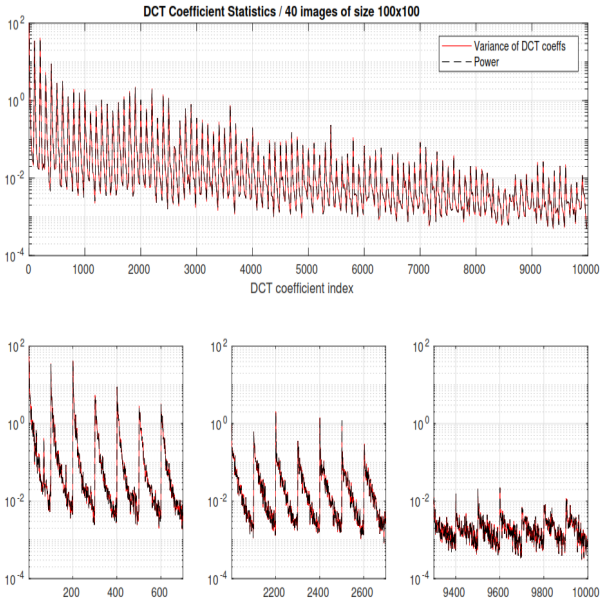


Fig. 2.8. *Top curve: DCT coefficients of an image. Bottom curves: zoomed version of the top curve [6].*

We can assume that coefficients follow a Gaussian distribution with mean of zero and variance σ_j^2 is contaminated itself in white Gaussian noise $w_i \sim \mathcal{N}(0,c)$. In this case the denoiser adapted to this signal model is calculated as [6]:

$$\begin{aligned} \hat{s}_j = F_j(u_j;c) = E_{s_j}(S_j|U_j = u_j) = \frac{\sigma_j^2}{\sigma_j^2 + c} u_j \\ F_j'(u_j;c) = \frac{d}{du_j} F_j(u_j;c) = \frac{\sigma_j^2}{\sigma_j^2 + c} \end{aligned} \tag{50}$$

The Figure 2.9 helps us compare the efficiency level of AMP and BAMP.

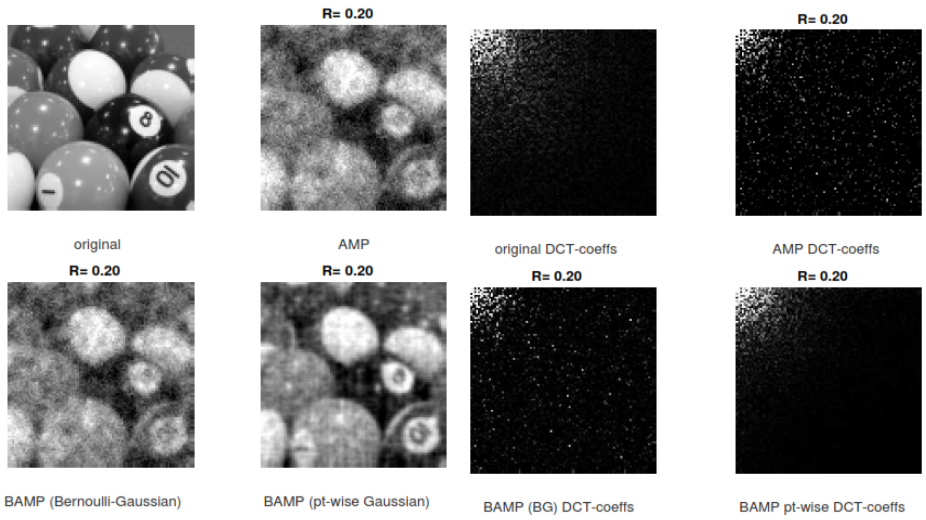


Fig. 2.9. a) The image on bottom left assumes that all the N entries of DCT coefficients follow the same Bernoulli-Gauss prior. The image on bottom left assumes that the DCT coefficients are point-wise Gaussian with different variance value for each entry in the DCT coefficinet matrix which is estimated from the data. As eis evident from images, point-wise Gaussian assumtion shows superior performance in comparison with its Bernoulli-Gauss counterpart. b) The recovered DCT coefficients for the image using BAMP with point-wise Gaussian assumption shows more accuracy in comparison with Bernoulli-Gauss assumption BAMP and also AMP [6].

As is evident from the figure, the DCT coefficients of the image are more accurately recovered when BAMP (point-wise) has been used. Consequently the BAMP recovered image has shows more detail and despite the great compression rate of .2, one can read the number on the balls! In fact this is immediately visible in the DCT coefficients. As discussed earlier, most of the energy in DCT coefficient matrix is concentrated in top left corner of the matrix. A glimpse on the DCT coefficient matrix recovered by Bernoulli-Gauss BAMP or AMP techniques reveals some white dots or larger values dispersed all over the recovered DCT coefficient matrix which should not be the case. The point-wise Gaussian BAMP yields a DCT coefficient matrix which is a better match to the original DCT coefficient matrix.

3 Foveated single-pixel imaging

We already noticed that the CS-based imaging system reduces considerably the sampling and also transmission bandwidth. In fact thanks to some sophisticated recovery schemes we can recover the images using a number of measurements much smaller than the number of pixels in the image which can be termed as sub-Nyquist sampling. Recently scientists have claimed that we can improve the compression rate even further

by mimicking the human visual system.

To come to a better understanding of what foveation single-pixel imaging means, I start with a figure which shows the anatomy of the human vision system:

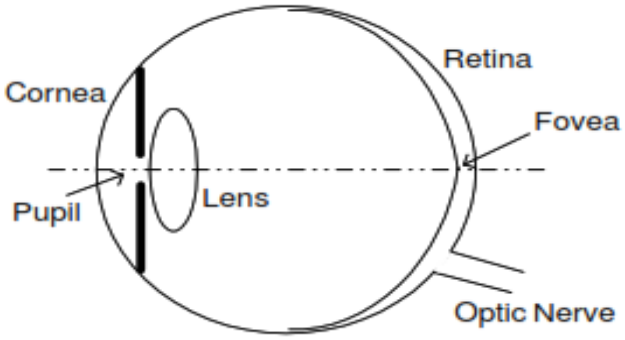


Fig. 3.1. *Human vision system and foveation [30].*

The light from the scene is passed by the lens into the fovea area. The fovea area has highest density of light sensors, namely photoreceptors. The density of these light sensors decreases exponentially towards the periphery with increasing eccentricity. This simply means that our visual system samples the area around the fixation point or center of our gaze with much higher resolution and the surrounding of the fixation point with much smaller resolution [29][30]. This motivated the scientists to mimic the human vision system in signal processing to obtain much higher resolution and consequently less recovery error in some area of interest in an image or the underlying scene while simultaneously allowing higher an higher recovery error or less resolution in rest of the areas. This way we achieve much higher compression ratios while maintaining the required resolution or detailed vision in the areas of interest. Now let's have a look on some technique employed in recent years to do foveation imaging which will be classified into two categories.

3.1 Superpixel method

Recall from previous section that in order to do CS-based sensing uniformly from the scene we employed a set of uniform random pixel grids which provided spatially constant resolution over the image. One way to mimic foveation is to use spatially variant pixel sizes or cells by combing or grouping the neighboring pixels together. Here we use cell with smaller size in the foveated region and use cells or superpixels with larger and larger size as we get distance from the foveated region into the margins of the vision field.

Figure 3.2 depicts a log map distribution of super pixels.

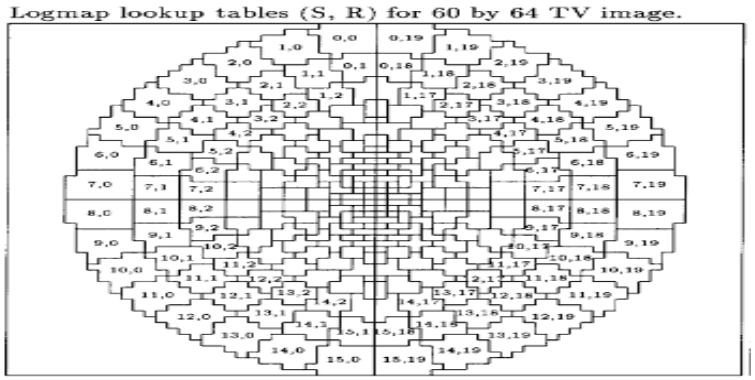


Fig. 3.2. *Logmap superpixel representation [30].*

One problem with above mentioned superpixel approach is that it is indeed difficult to realize. A simpler technique is to use rectangular shaped cells. This way it's easier to implement.

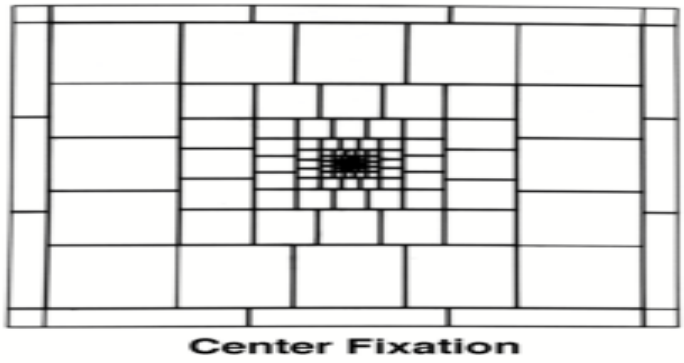


Fig. 3.3. *Superpixel using concentric rectangular rings [31].*

The superpixel technique is employed during the sampling or sensing of the image and the recovery process can be left as is. In other words the employed recovery technique is blind to the distribution of the pixels and their sizes and treats the sampled image as if uniform sampling pattern with constant spatial resolution had been employed to do sampling.

First row in the Figure 3.4 depicts the uniform pixel grid and the random patterns used for CS-based uniform sampling from the original image. Second row exploits nonuniform pixel grid and as a result, nonuniform random patterns for CS-based nonuniform or foveated sampling from the image. As is evident from the image, nonuniform pixel grid uses the superpixel ideology. The pixels located in the center have small size which translates into higher resolution for the central part of the image. The surrounding area represents a logmap superpixel distribution.

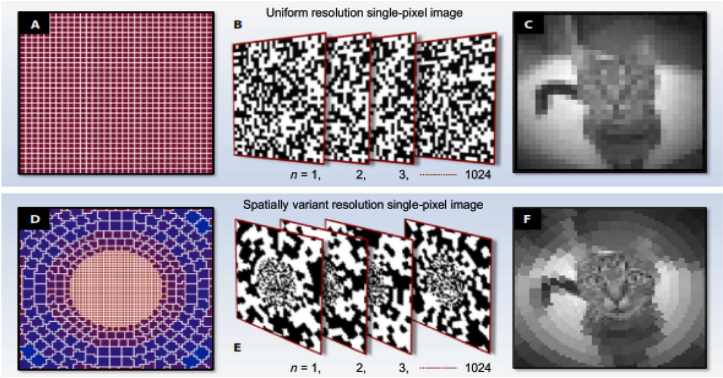


Fig. 3.4. *a) Uniform pixel grid. b) Random patterns based on Hadamard basis. c) Recovered image of a cat. d) Nonuniform pixel grid e) Nonuniform random patterns. f) Recovered image of cat as a result of foveated imaging [32].*

As one can be seen from the figure above, as a result of smaller sized super pixels in the center of the image, one can see more details in the center than the margins.

3.2 Foveation operator or filtering method

In previous section foveated imaging using superpixel method was discussed. This section concentrates on other foveated imaging techniques which exploit filtering and foveation operator to introduce focusing in an image. In what follows one can find some details regarding these techniques.

3.2.1 Linear shift variant foveation filter

The first approach uses a bank of low-pass filters each of which determines the pixel value in the recovered image based on required resolution.

The above mentioned approach is equivalent to using a linear shift variant foveation filter (low-pass) which operates on every pixel of the underlying image and whose cut-off frequency $w_c(u,v)$ varies dependent on the location of the pixel (u,v) [34]. For instance, the cutoff frequency value of 1.0 is considered for the foveation regions (which means no filtering) and cutoff values less than 1.0 or close to zero for those pixels far from the foveation center which translates into stronger low-pass filtering operation for the pixels as they get distant from foveation center.

Because the foveation filter is linear it can be modeled as [34]:

$$F = \begin{bmatrix} L_{w_c(0,0)}(0,0) \\ L_{w_c(0,1)}(0,1) \\ \vdots \\ L_{w_c(0,N-1)}(0,N-1) \end{bmatrix} \tag{51}$$

here, each row represents a 2D [2] low-pass FIR filter with cutoff frequency which is padded and shifted to be centred at pixel (u,v).

By applying this filter to the original image we get the foveated image [34]:

$$\mathbf{X}^{Fov} = \boldsymbol{F} \times \mathbf{X} \tag{52}$$

Now by combining the compressed sensing and foveation filtering we have [34]:

$$\mathbf{y} = \boldsymbol{\Phi} \times \mathbf{s} = \boldsymbol{\Phi} \times \boldsymbol{\Psi}^{\star} \times \mathbf{x} = \boldsymbol{\Phi} \times \boldsymbol{\Psi}^{\star} \times \mathbf{f} \times \mathbf{x} \tag{53}$$

Where the $\boldsymbol{\Psi}^{\star}$ matrix represents the direct sparsifying transform like DWT or DCT.

One of the key features of the compressed sensing is that it places low computational complexity on the sensor or transmitter side. In order to stay tuned with this feature we can simply precompute the resulting sampling matrix and apply it on the original image.

The Figure 3.5 displays two examples of the foveation by applying the linear shift variant low-pass filter:

(a) **(b)**

Fig. 3.5. *Recovered images using by linear shift variant low-pass filtering **a)** Foveated bridge image. **b)** Foveated Lena's image [34].

Both images exhibit higher resolution in the foveation center.

[2] Two dimensional

3.2.2 DWT based foveation operator

One other way to create foveated images is to bring the original signal to the wavelet domain by applying discrete wavelet transform and then applying a mask to the resulting wavelet coefficients and finally recover the foveated image by applying inverse DWT.

In general a typical one dimensional mask can be of the form [35]

$$(Tf)(x) := \int_{-\infty}^{\infty} f(t)\frac{1}{w(x)}g(\frac{t-x}{w(x)})dt \tag{54}$$

$$w = \alpha|x-\gamma|+\beta \tag{55}$$

Where w is the weighing function and g is a Gaussian smoothing function. In the weighing fuction formula, the value α, β and γ determine the resolution descrease rate, foveal resolution and fovea location respectively. The Equations 54 and 55 can be rewritten as [35]

$$(Tf)(x) = f_f = DWT^{-1}\{mask \times DWT(f)\} \tag{56}$$

Where *mask* represnets the DWT of the kernel function which defines the integral in the Equation 54. The matrix mask introduces foveation into the wavelet domain.

This foveation technique can be simply generalized to 2D case by using 2D smoothing and weighting functions in the Equation 54.

The operations above have nothing to do with compressed sensing and they are just an example how we can bring foveation into an image using its wavelet representation. It's interesting that we can go one step forward and apply compressed sensing to this scheme. Now we combine the compressed sensing with foveation technique above to gain compression and foveation simultaneously. This combination can be formulated as [35]:

$$\mathbf{y} = \mathbf{\Phi}\times\mathbf{x}^{Fov} = \mathbf{\Phi}\times\mathbf{\Psi}\times\mathbf{M}\times\mathbf{s} = \underbrace{\mathbf{\Phi}\times\mathbf{\Psi}\times\mathbf{M}\times\mathbf{\Psi}^{\star}}_{\mathbf{A}}\times\mathbf{x} = \mathbf{A}\times\mathbf{x} \tag{57}$$

Here the Ψ is the $N\times N$ orthonormal basis, $\Psi^{\star}$ is the direct DWT transform and $M = diag\{mask\}$. Here the diagonal elements of the *mask* can be approximated and very small values can be approximated with zero which simplifies things and the multiplication process and the resulting foveated image then becomes an approximation.

Basically what the Equation 57 means is that forming the composite sensing matrix $\mathbf{A}$ and applying it to the original signal is equivalet to applying the primitive sensing matrix Φ to the unavailable foveated version of the image. The recovery procedure yields the sparsest solution to following optimization problem [35]:

$$\min_{\mathbf{s}} \left\| \underbrace{\mathbf{M}\times\mathbf{s}}_{\mathbf{s}^{Fov}} \right\|_0 \quad s.t. \quad \mathbf{y} = \mathbf{\Phi}\times\mathbf{x}^{Fov} = \mathbf{\Phi}\times\mathbf{\Psi}\times\mathbf{M}\times\mathbf{s} = \mathbf{\Phi}\times\mathbf{\Psi}\times\mathbf{M}\times\mathbf{\Psi}^{\star}\times\mathbf{x} \tag{58}$$

The composite sensing matrix helps us get the foveated version of the wavelet coefficent vector after the recovery process. Then, inverse DWT can be applied to the foveated coefficient vector to yield the compressively sensed foveated image. The optimization problem in Equation 58 can be solved by employing any of the recovery algorithms described in previous sections. The matrix used for recovery in this case will be $\Phi \times \Psi$ and the solution will be the sparse masked coefficient vector s^{Fov} which after inverse wavelet transform yields the foveated image. As we already know, in order for the optimization problem to have correct solution, the matrix $\Phi \times \Psi$ must satisfy the RIP property. In fact we have already designed the matrix Φ to meet this criteria. As far as the matrix Φ is an i.i.d Gaussian matrix and the matrix Ψ is orthogonal, the resulting $\Phi \times \Psi$ is again i.i.d Gaussian. This ensures that our recovery procedure is successful. Since matrix M is not an orthogonal matrix, the composite matrix $\Phi \times \Psi \times \boldsymbol{M}$ does not always satisfy the RIP requirement and hence it can not be used for recovery purpose.

If we further assume that we are directly sensing the wavelet coefficients the formula can be rewritten as [35]:

$$\mathbf{y} = \boldsymbol{\Phi} \times \mathbf{s}^{Fov} = \boldsymbol{\Phi} \times \mathbf{M} \times \mathbf{s} = \underbrace{\boldsymbol{\Phi} \times \mathbf{M} \times \boldsymbol{\psi}^{\star}}_{\mathbf{A}} \times \mathbf{x} = \mathbf{A} \times \mathbf{x} \tag{59}$$

And the recovery procedure yields the sparsest solution to following optimization problem [35]:

$$\min_{\mathbf{s}} \left\| \underbrace{\mathbf{M} \times \mathbf{s}}_{\mathbf{s}^{Fov}} \right\|_0 \quad s.t. \quad \mathbf{y} = \boldsymbol{\Phi} \times \boldsymbol{s}^{Fov} = \boldsymbol{\Phi} \times \mathbf{M} \times \mathbf{s} = \boldsymbol{\Phi} \times \mathbf{M} \times \boldsymbol{\Psi}^{\star} \times \mathbf{x} \tag{60}$$

This optimization problem can be solved by employing any of the recovey algorithms described in previous sections. The matrix used for recovery in this case will be Φ and the solution will be the sparse masked coefficient vector s^{Fov} which after inverse wavelet transform yields the foveated image. As we already know in order for the optimization problem to have correct solution, the matrix Φ must satisfy the RIP property. In fact we have already designed the matrix Φ to meet this criteria. As far as the matrix Φ is an i.i.d Gaussian matrix for instance we are secure. If we use the matrix $\Psi \times \boldsymbol{M}$ for recovery purpose the RIP criteria is not satisfied anymore. Because M is not an orthogonal matrix.

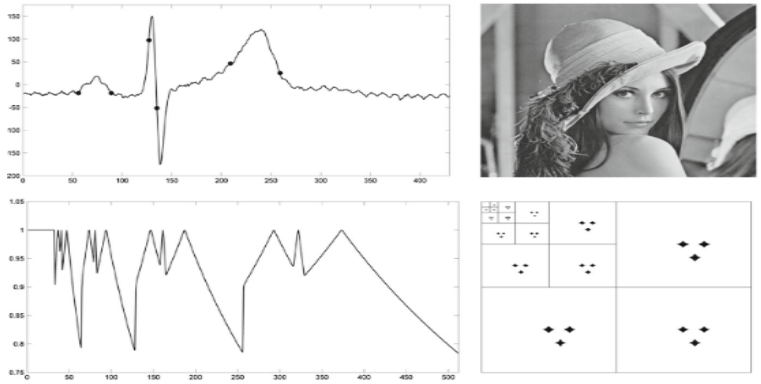


Fig. 3.6. *Top left: The Electrocardiogram (ECG) signal, Bottom left: The one dimensional foveation mask for ECG signal. Top right: Lena's image. Bottom right: The two dimensional foveation mask for the image. The foveation points placed on cneter of the eyes and the tip of the nose [35].*

The Figure 3.6 demonstrates the foveation mask for two signals. First signal is the critical one dimensional ECG signal and the second one is a 2D signal, namely Lena's image. These masks help us recover some parts or areas of an image or a signal with higher resolution.

It's worthwhile to monetion that foveated imaging techniques described above, provide foveation at the sampling or measurement stage. There are some other techniques that involve both sampling and reconstruction or recovery stages which is out of scope of this thesis.

4 Single-pixel imaging and performance analysis

In previous chapters the thesis presented an overview on some interesting CS-based imaging approaches. Some common approaches to uniform and foveated single-pixel imaging were introduced. This chapter presents a typical single-pixel imaging system implemented in Matlab which deploys both unfoveated and foveated single-pixel imaging. In terms of novelty it introduces patch-based and superpixel-based focusing techniques which fall into the category of foveated single-pixel imaging.

The implemented single-pixel imaging system seeks to answer three critical questions in the area of single-pixel imaging based on simulations and numerical analysis. First, the thesis seeks to answer whether the recommended patch-based focusing technique introduces some kind of artifact to the recovered image. Second, to study if the recommended focusing technique performs better than its superpixel-based counterpart in terms of the quality or signal to noise ratio of the recovered images. Third, the thesis studies whether there are some random sensing matrices which perform better than others to check whether random designs differ significantly from each other or their performance is close to the average which is practically very important.

4.1 Implemented single-pixel imaging system in Matlab in a nutshell

As can be seen from the Figure 4.1, the implemented system starts with producing the DCT basis matrix which will be further used to sparsify our original image. In other words the image is first transformed to a sparse DCT coefficient domain. Before the DCT coefficient matrix is input to the CS-based sampling or measurement unit, it is vectorized using Matlab 'vec' operator. Please consult the Subsection 2.3 for more detail regarding the DCT transform on images.

The implemented code follows with creating suitable sensing matrix which in this case in a i.i.d Gaussian matrix which ensures indeed the RIP property. This matrix is generated in uniform and nonuniform formats depending on our choice. The nonuniform format helps us bring foveation or focusing in the image by employing the superpixel ideology explained earlier. Next, the sensing matrix is applied to the sparse vectorzied version of the DCT coefficients corresponding to the original image. Finally the resulting measurement vector is fed into the recovery unit to solve the optimization problem explained earlier. The implemented recovery methods are: LASSO, AMP and BAMP. The resulting sparse solution is again transformed back into the matrix form before the inverse DCT transform is applied to it to yield the approximated image.

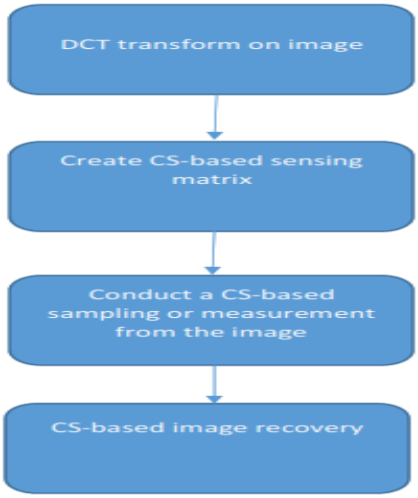


Fig. 4.1. *Flowchart for the implemented single-pixel imaging system*

4.2 Implemented CS-based focusing techniques

As explained in previous sections, there are several ways one can introduce foveation into the CS-based digital imaging. Please recall from previous chapters that CS-based sampling is a series of inner product of the underlying scene and random patterns. One way to mimic foveation is to use patterns with spatially variant pixel sizes or cells (nonuniform patterns) by combining or grouping the neighboring pixels together. Here we use cells with smaller size in the foveated region and use cells or superpixels with larger and larger size as we get distant from the foveated region and approach the margins of vision field. In other words we employ a nonuniform or spatially variant pixel grid to generate random patterns which will be used further for sampling or inner product operation explained earlier. Henceforth, for the sake of simplicity, I will use the term superpixel-based focusing to refer to this method.

In another approach the image is sampled using uniform patterns with spatially equal pixel sizes but with different rate for different patches of the image. In other words, the image is divided into patches of equal dimension and foveation is gained by investing higher rate for the important patches and lower rate for the ones which are not important. Here the important patches are the ones which are located in the areas within the image which higher resolution or detail is required. Henceforth, for the sake of simplicity, I will use the term patch-based focusing to refer to this method.

In what follows, two main methods employed by Matlab code to implement focusing are discussed. Afterwards a performance analysis is run to compare the efficiency level of the implemented techniques.

4.2.1 Superpixel-based focusing

This method exploits only one patch to conduct a CS-based sampling from the underlying image. In general there are five variants of these matrices used in this implementation. One drawback of this method is that creating a random nonuniform patch to use in the sensing stage is computationally time consuming.

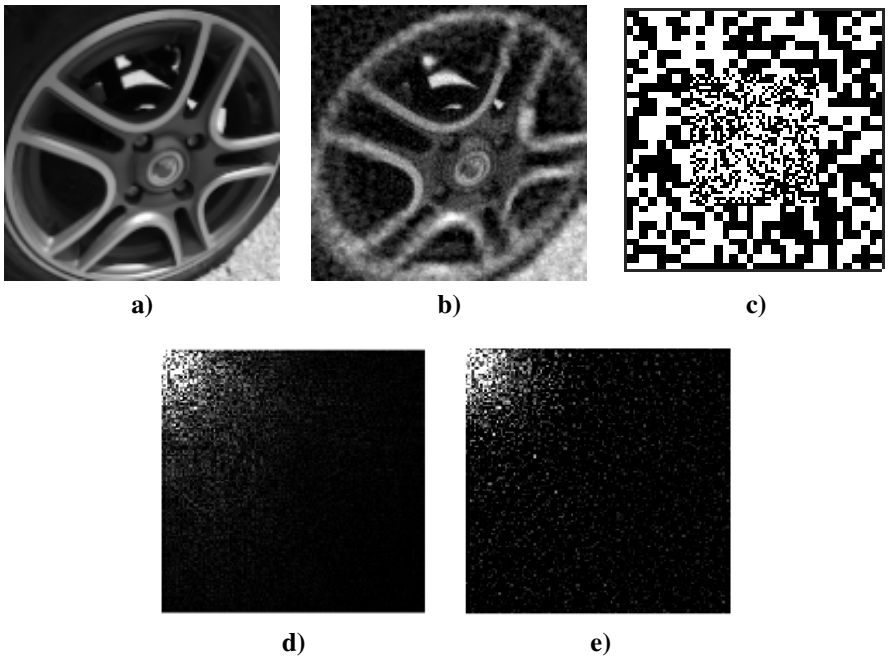


Fig. 4.2. **a)** *Original image.* ***b)*** *Recovered image.* ***c)*** *Non-uniform sensing matrix.* ***d)*** *Original DCT coefficients.* ***e)*** *Recovered DCT coefficients.*

For the above-mentioned superpixel-based method, BAMP recovery algorithm does not converge and this is because the assumption for the sensing matrices made in the BAMP derivation are not fulfilled. This includes in particular that not all matrix elements can be roughly approximated by $A_{ij}^2 \approx 1/M$ with M the number of measurements taken. Consequently, the LASSO recovery scheme was deployed for the superpixel-based method.

4.2.2 Patch-based focusing

As mentioned earlier, another focusing technique is also introduced in this work which divides the image into the non-overlapping patches of the same dimension and proceeds with the CS-based sensing and recovery of each patch individually. This is unlike the superpixel-based focusing in that it operates on whole image at once using a nonuniform sampling matrix or pattern, The foveation is gained by investing higher sampling ratio for the patches which spatially are located in points of interest where more resolution is at high demand. This thesis has assumed that the center of image is the point of high interest or focus. The final large image is then formed using assembling differ-

ent patches together in a fashion quite like putting pieces of puzzle together. One big advantage of this method is that it's much speedier than its counterpart when it comes to Matlab execution time because of two reasons. First, as a result of dividing the large image into patches, the implemented sampling and recovery operate on matrices of smaller size which considerably lowers the execution time for matrix multiplication for instance. Second, the creation of uniform matrices are considerably less time consuming in comparison with nonuniform matrices of the same dimension.

As already discussed, One of major concerns of the thesis is to check whether using the patch-based focusing, which involves individual processing of the patches, leads to emergence of artifacts to the final recovered image or not. Simulations show that there is no need to worry about this issue and the effect of artifacts is negligible. Intuitively, the effect of artifacts can be visible when smaller and smaller values are invested for the recovery of patches or when the noise becomes stronger. In such cases one also can use overlapping patches in the patch margins and then to do pixel-averaging to get better results in the overlap regions.

The Figure 4.3 below exemplifies a typical recovered image as a result of applying patch-based focusing technique to the underlying image. It uses a sampling ratio of $R =$ 0.3 for the central part and a ratio of R = 0.2, 0.1 and .09 for the surrounding patches as they distance themselves from the center. It exploits BAMP recovery method.

(a) **(b)**

Fig. 4.3. ***a)*** *Original image.* ***b)*** *Recovered foveated image using patch-based method.*

Now let's have a look into some patches inside the image above and study the visual acuity and effect of sampling ratio on the accuracy of the BAMP recovery algorithm and consequently the resolution of the recovered image.

The Figure 4.4 demonstrates one of the central patches which received high sampling rate $R = 0.3$. As one can see the recovered patch has fine resolution and the DCT coefficients are recovered with acceptable precision.

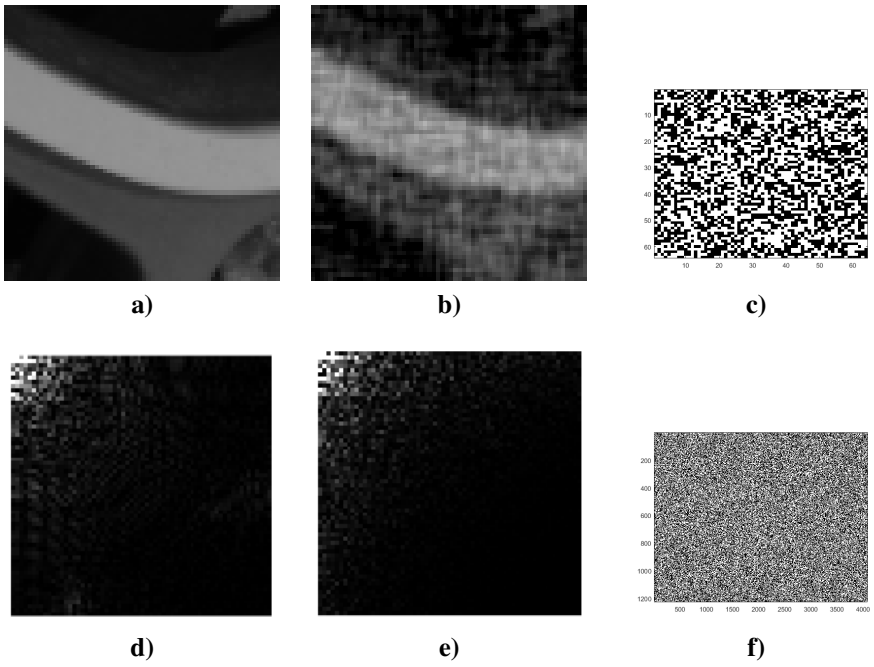


Fig. 4.4. ***a)*** *Original image.* ***b)*** *Recovered image.* ***c)*** *Uniform sensing pattern or matrix.* ***d)*** *Original DCT coefficients.* ***e)*** *Recovered DCT coefficients.* ***f)*** *Sensing matrix.*

One of the sampling patterns is shown in the Figure 4.4c. As is evident from the uniform pattern, the cells or pixels are spatially of the same size. The dimension of the patterns and the patch from the original image are identical.

The sensing matrix is shown in the Figure 4.4f. Each row of the sensing matrix is in fact one of the sampling patterns that are stacked together horizontally. Intuitively the number of columns in the sensing matrix is equal to the number of pixels in the patch and the number of rows is equal to the number of times we have acquired samples from the underlying patch. By dividing the number of rows in the sensing matrix by the number of columns we get the CS-based sampling rate R = 0.3 which was used to sense and recover the corresponding patch.

As can be seen from the Figure 4.5, one of the patches near the image border which received the smallest sampling rate and consequently least recovery accuracy and resolution inside the recovered patch.

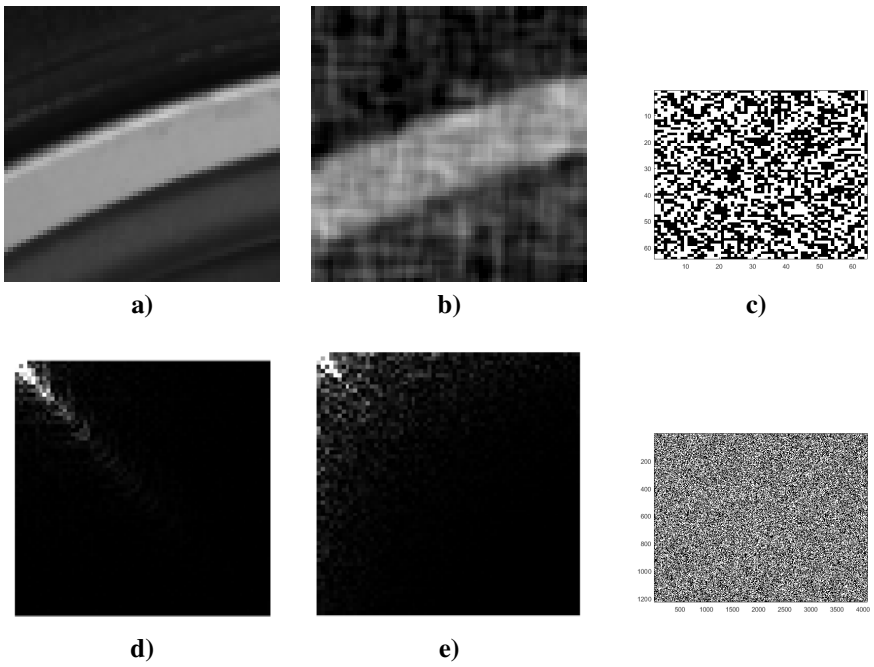


Fig. 4.5. ***(a)*** *Original image.* ***b)*** *Recovered image.* ***c)*** *Uniform sensing matrix.* ***d)*** *Original DCT coefficients.* ***e)*** *Recovered DCT coefficients.* ***f)*** *Sensing matrix.*

Intuitively, lower sampling rate for this patch leads to worse performance in estimating the DCT matrix and thus higher discrepancy between the original DCT matrix of the patch and that of recovered one. The Figure 4.5c shows one of the sampling patterns used. As before, the dimension of the pattern is identical to the original patch from the image. Figure 4.5f shows the sensing matrix to recover this patch in general. As discussed above, by dividing the number of rows of the matrix to the number of columns we get the CS-based sampling rate R = 0.09 which was used to sense and recover the corresponding patch.

For the patch-based focusing or sampling technique explained above, one can use LASSO, AMP and BAMP recovery algorithms. For the reasons explained earlier in this thesis, BAMP algorithm demonstrates superior performance. It does not converge for the superpixel-based method but it converges and in fact does well in terms of the precision and quality of the recovered image. Consequently, although the performance of recommended patch-based method combined with all three implemented recovery methods are studies in following section, patch-based method employing the BAMP recovery is the method of choice.

4.2.3 Performance comparison between suggested focusing techniques

Now it's time to target the second question, namely whether the patch-based foveated imaging technique does better than the nonuniform or superpixel-based focusing in terms of quality or SNR [3] of the recovered images.

To test that, a database of 52 images were compressively sensed using above-mentioned focusing techniques and recovered using implemented recovery schemes (LASSO, AMP, BAMP) and subsequently their signal to noise ratio are computed and exploited to get the mean and confidence interval (CI) of our measurement using the Formulas 61, 62 and 63. The corresponding mean and CI are afterwards used as basis to determine which focusing technique and which recovery method together give best performance.

The formulas below (with $n = 52$) are used to compute the mean and confidence interval (CI) from the SNR of 52 recovered images:

$$\overline{x_n} = \frac{\sum_{i=1}^{n}(x_i)}{n} \tag{61}$$

$$\sigma_n = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x_n})^2} \tag{62}$$

$$c_n = 1.96\frac{\sigma_n}{\sqrt{n}} \tag{63}$$

Where x_i represents the signal to noise ratio for the i'th image. Parameters $\overline{x_n}$ and σ_n represent the mean and standard deviation up to the n'th image. Finally c_n represents the 95 confidence interval up to n'th image.

As already discussed, among the implemented recovery schemes, only the LASSO functioned well for the nonuniform or superpixel focusing and the other recovery schemes (AMP and BAMP) did not converge. In what follows, one can find the simulation results regarding the performance comparison of the super-pixel-based sampling (using LASSO) and the patch-based sampling (using BAMP, AMP and LASSO).

Please consult the Table 4.1 where the simulation options and simulation results are shown. In this table superpixel-based sampling is compared against the patch-based sampling with various recovery schemes.

[3] Signal to noise ratio

Simulation options and results					
Focusing method (Recovery scheme)	Image size	Number of patches	Patch size	Total rate	SNR Mean $\pm$ CI
Superpixel-based focusing (LASSO)	128×128	1	128×128	0.2	14.14 ± 0.70
Patch-based focusing (BAMP)	128×128	4×4	32×32	0.2	13.16 ± 0.67
Patch-based focusing patch sampling (AMP)	128×128	4×4	32×32	0.2	9.12 ± 0.72
Patch-based focusing (LASSO)	128×128	4×4	32×32	0.2	10.32 ± 0.71

Table 4.1: *Simulation options and results for performance comparison between patch-based method (using BAMP, AMP and LASSO) vs. superpixel-based method (using LASSO).*

As can be seen from the Table 4.1, the super-pixel focusing (combined with LASSO recovery) shows performance of around 14.14 dB which is approximately 1 dB better than patch-based method employing BAMP recovery. For the patch-based focusing method the 128 images and patch size of 32×32 were selected in order to give 16 patches. This way larger rate investment could be done in the central patches for higher resolution in the center of image. Larger image sizes could be used to do the simulation but the number of patches and also the resulting patch dimensions would rise which is identical to much higher simulation time for Matlab.

In fact this result is practically very important. Although the super-pixel based method has slightly better performance in terms of precision in recovery of compressively sensed images, it is computationally much more time consuming than the patch-based focusing recommended by this research. The superpixel-based technique is computationally complex and intensive at random sampling matrix generation and specially recovery steps. The principle reason is that in superpixel-based technique we have only one patch which is actually with dimensional of the original image under sample. Consequently, all the underlying sampling and sensing and subsequent recovery procedures involve operations on vectors and matrices of considerably much larger size which slows down the Matlab's processing speed. For instance for the pixel-based method, the average execution time for the BAMP recovery in Matlab is around $4 \times 10^{-4}(s)$ while the execution time for the LASSO recovery in superpixel-based method is around $0.7(s)$. Thus, one can ignore the 1 dB improvement of superpixel method and employ the patch-based method combined with BAMP recovery to be able to be faster and in fact to offer best compromise between speed and performance in terms of quality or SNR or recovered images.

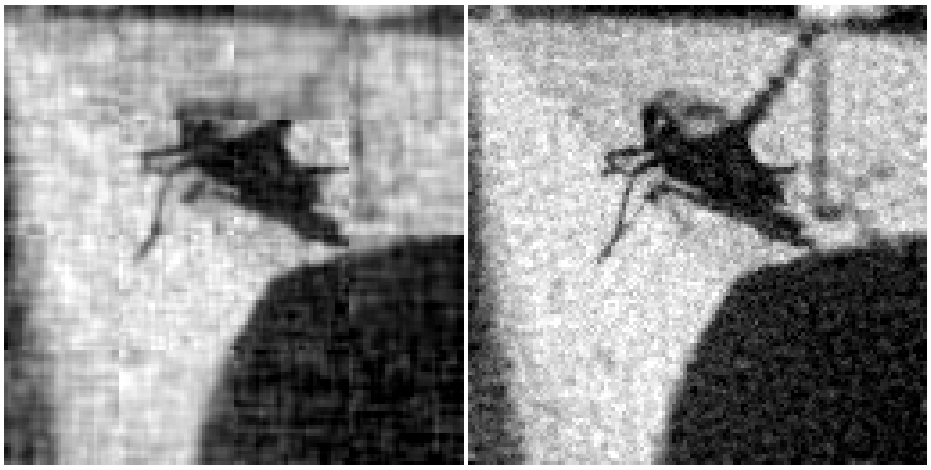
As discussed earlier, patch-based focusing operates on non-overlapping patches of the image and invests larger rate for the patches of higher importance and lower rate for other patches. The resulting overall rate for the image can is then calculated as:

$$r_t = \frac{m \times r_h + n \times r_l}{m+n} = \frac{4 \times 0.5 + 12 \times 0.1}{4+12} = 0.2 \tag{64}$$

Where parameters m and n are the number of high quality and low quality patches

respectively. The parameters r_h and r_l represent the rate for the high resolution and low resolution patches respectively.

The Figure 4.6 depicts the recovered images using patch-based and superpixel-based focusing methods:



(a) **(b)**

Fig. 4.6. ***a)*** *Recovered image using patch-based method (using BAMP recovery).* ***b)*** *Recovered image using superpixel-based method.*

In the Figure 4.6a one can observe that the 4 central patches have higher resolution than the other patches which is because of larger rate (R = 0.5) being invested for those patches and smaller rate (R = 0.1) for other patches. The central part of image 4.6b has higher resolution as a result of smaller superpixels in the center of the nonuniform pixel grid used for sampling the image.

Please consult the Figure 4.7 for a graphical demonstration of the performance comparison between patch-base focusing (using BAMP) and the superpixel-based focusing (using LASSO) techniques.

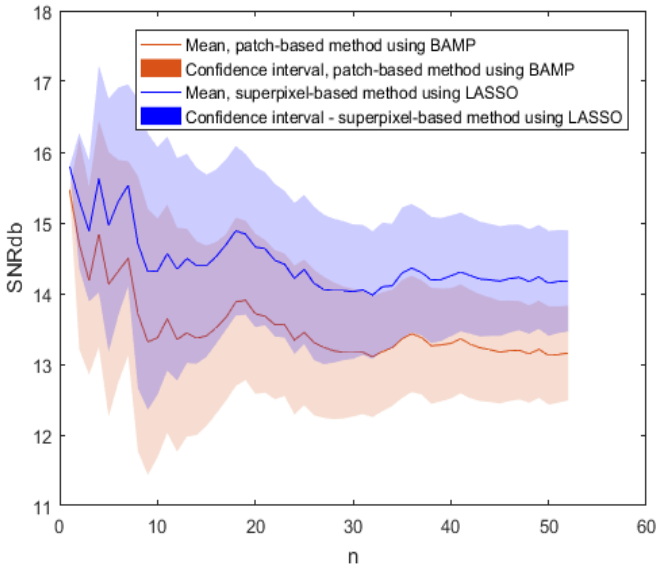


Fig. 4.7. *Performance comparison: Patch-based focusing (BAMP) vs. superpixel-based focusing(LASSO)*

In order to draw the curves for the Figure 4.7, the Matlab code computes the mean and the confidence interval up to n'th image using Formulas 61, 62 and 63 to show progressively how the mean and confidence interval values evolve as the simulation proceeds to next images. As is apparent from the Figure 4.7, as the simulation proceeds to include larger number of images, the mean SNR for the superpixel-based method converges to value 14.14 dB which is equal to averaged SNR values for 52 images in the image database. Also on can observe from the image that patch-based method using BAMP approaches mean SNR of around 13.16 which means the super-pixel method performs better.

Please consult the Figure 4.8 for a graphical demonstration of the performance comparison between patch-base focusing (using AMP recovery) and the superpixel-based (using LASSO recovery) focusing techniques.

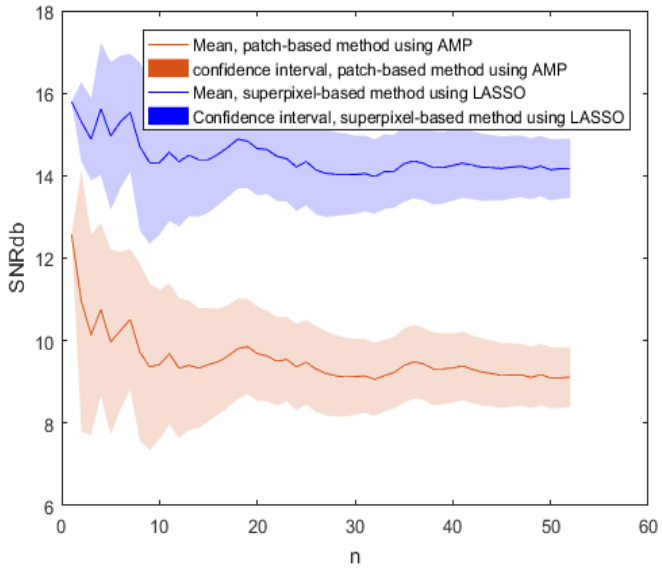


Fig. 4.8. *Performance comparison: Patch-based focusing (AMP) vs. superpixel-based focusing(LASSO)*

Again, in order to draw the curves for the Figure 4.8, the Matlab code computes the mean and the confidence interval up to n'th image using Formulas 61, 62 and 63 to show progressively how the mean and confidence interval values evolve as the simulation proceeds to next images.

As is apparent from the Figure 4.8, as the simulation proceeds to include larger number of images, the mean SNR for the superpixel-based method converges to value 9.12 dB . This means that superpixel-based method performs around 5 dB better than patch-based method (using AMP).

Please consult the Figure 4.9 for a graphical demonstration of the performance comparison between patch-base focusing (using LASSO) and the superpixel-based (using LASSO) focusing techniques.

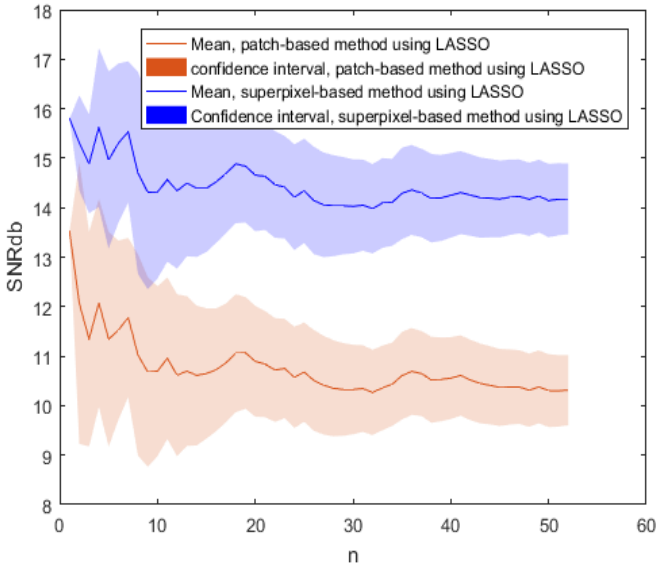


Fig. 4.9. *Performance comparison: Patch-based focusing (LASSO) vs. superpixel-based focusing(LASSO)*

Again, in order to draw the curves for the Figure 4.7, the Matlab code computes the mean and the confidence interval up to n'th image using Formulas 61, 62 and 63 to show progressively how the mean and confidence interval values evolve as the simulation proceeds to next images.

As is apparent from the Figure 4.9, as the simulation proceeds to include larger number of images, the mean SNR for the superpixel-based method converges to value 10.32 dB . This means that superpixel-based method performs around 3.8 dB better than patch-based method (using LASSO).

4.3 Sensing matrix design

Now that the answer to first two questions are already given, let's try to concentrate on the third question, namely does any single random design for the sampling or sensing matrix has a performance close to the performance averaged over many sensing matrices or there are some random matrices which have considerably better performance. In the implemented single-pixel imaging system one matrix is picked randomly among many possible random i.i.d Gaussian matrices. In case the answer to the above-mentioned question is positive, we need to worry about the design and in order to make improvement to the overall system performance and be selective in choosing the right sensing matrix. The answer to this question is practically very important because in practice one needs to deploy a fixed sensing matrix which could be computed at initialization stage and live with it throughout its normal operation.

The implemented Matlab code uses a series of two main cycles (C1 and C2). Each cycle iterates through a number of 52 images of dimension 128×128 from the image databse and performs uniform CS-based sensing and recovery on them.

C1 and C2 cycles are different in that C1 uses a fixed single sensing matrix for all 52 images from the database. In contrast, C2 cycle changes the sensing matrix image to image. Cycle C1 is executed 10 times, each time with a new choice for the single fixed sensing matrix. Cycle C2 is executed 1 time. At the end of each C1 execution round as well as single C1 execution round, the mean and confidence interval for the SNR of the recovered images calculated using Formulas Formulas 61, 62 and 63 with setting parameter with $n = 52$.

The mean SNR and confidence interval calculated for C2 cycle represents the performance averaged over many (52) sensing matrices.

The mean SNR and confidence interval calculated for each of C1 execution rounds represents the performance for the given single fixed sensing matrix in that round. C1 cycle is executed 10 times because the simulation wants to exclude the possibility of being lucky in the choice of the single fixed matrix.

The simulation also takes the effect of additive Gaussian noise in the sensing step into account. Among implemented recovery schemes, BAMP is used for this round of simulation simply because it demonstrates superior performance in comparison with AMP and LASSO. Furthermore, the sampling rate of 0.2 is chosen. Larger or smaller values for the rate was possible. But the rate chosen is a appropriate compromise between simulation speed and the quality of recovered images.

1st cycle (C1):
This cycle compressively senses a bunch of 52 images with dimension 128×128 uniformly using a single fixed sensing matrix. As the name 'uniform' implies, here no focusing is done and the whole image is seen as one patch and whole image is sensed using a simple rate. Then Matlab code is instructed to do this cycle 10 times and every time it changes the fixed signle sensing matrix to test that any single fixed random design has a performance close to average performance which is computed in the second cycle (C2).

2nd cycle (C2):
The second cycle is quite like the first cycle (C1) with the difference that the sensing matrix varies image to image to enable us to calculate the average performance. This cycle is executed only 1 time.

The table 4.2 summarizes the simulation options and the performance for each of ten C1 cycles as well as the single C2 cycle.

Simulation options and results					
Cycle type - iteration number	SNR (dB) Mean ± CI	Recovery scheme	Rate	Image size	Patch size
C1 - 1	13.99 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 2	13.95 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 3	13.97 ± 0.63	BAMP	0.2	128×128	128×128
C1 - 4	14.03 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 5	14.00 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 6	14.00 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 7	14.04 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 8	13.98 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 9	13.98 ± 0.62	BAMP	0.2	128×128	128×128
C1 - 10	14.01 ± 0.62	BAMP	0.2	128×128	128×128
C2 - 1	14.01 ± 0.61	BAMP	0.2	128×128	128×128

Table 4.2: *Simulation options and results for the cycles C1 and C2.*

Again the Formulas 61, 62 and 63 with $n = 52$ is used to calculate the mean and confidence interval from the SNR values. The Table 4.2 clearly shows that both C1 and C2 cycles demonstrate virtually the same performance. As cab be seen from the table, every choice for the fixed single sensing matrix (each C1 execution round) has a performance so close to the performance averaged over many sensing matrices (single C2 cycle execution round).

Please consult Figure 4.10 for a graphical demonstration of the performance comparison between the individual C2 cycle and one of the C1 cycle execution round.

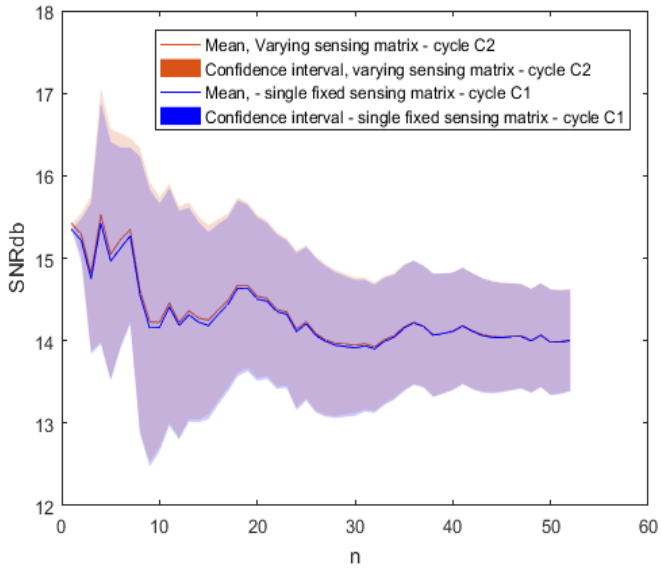


Fig. 4.10. *Image recovery performance comparison between the individual C2 and one of the C1 cycle execution rounds.*

In order to draw the curves for the Figure 4.10, the Matlab code computes the mean and the confidence interval starting from the first image up to the n'th image to show progressively how the mean and confidence interval values evolve as the simulation proceeds to next images according to the Formulas 61, 62 and 63.

As one can observe from the Figure 4.10, the mean and the confidence interval is approximately a match. In other words the performance remains unchanged irrespective of applying the single fixed sensing matrix or varying sensing matrix. This means that one single fixed sensing matrix provides the performance close to the performance averaged over many sensing matrices.

4.4 Conclusion

The primary goal of the thesis was to provide answers to some of the questions in the world of CS-based foveated digital imaging based on numerical analysis and simulations. It started with an introduction to single-pixel imaging. Then it provided the reader with current researches regarding unfoveated digital imaging. Next it put one step forward and studied some methods which help us introduce foveated imaging into the world of CS-based digital imaging.

Next, it explains the implemented single-pixel imaging system in Matlab and introduces patch-based and superpixel-based focusing methods to introduce focusing in images. Using simulation results, the thesis concludes that despite the patch-based focusing has slightly lower performance than the superpixel-based method, it's indeed much speedier in terms of implementation and execution time so it can be the method of choice.

This work also made it clear that as a result of the recommended patch-based focusing to create foveation virtually no artifact is introduced to the recovered image. Intuitively, the effect of artifacts can be visible when smaller and smaller values are invested for the recovery of patches or when the noise becomes more powerful. In such cases one also can use overlapping patches in the patch margins and then to do pixel-averaging to get better results in the overlap regions.

Last but not the least, at sensing matrix design stage, one needs not to worry about the choice of sensing matrix. In other words any single and fixed random sensing matrix has a performance completely close to the performance averaged over many sensing matrices and they do not differ in terms of performance. The answer to this question is practically very important because in practice only one fixed sensing matrix is used.

Abbreviations

RIP	Restricted Isotropic Property
CS	Compressed Sensing
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
DMD	Digital Micromirror Device
IST	Iterative Soft Thresholding
IHT	Iterative Hard Thresholding
AMP	Approximate Message Mapping
BAMP	Bayesian Approximate Message Mapping
CCD	Charge Coupled Device
CI	Confidence Interval
2D	Two dimensional

References

[1] *Weik M.H. (2000) Nyquist criterion. In: Computer Science and Communications Dictionary. Springer, Boston, MA*

[2] *G. Baraniuk, Richard. (2007). A lecture on compressive sensing. IEEE Signal Processing Magazine. 24. 1-9.*

[3] *R. Gonzalez, "Digital Image Processing," Chap 7 (Wavelet transforms), Chap 8 (Image compression)*

[4] *E.-C. Chang, S. Mallat & C. Yap, "Wavelet foveation," Journal of Applied and Computational Harmonic Analysis, vol. 9, no. 3, pp. 312-335, Oct. 2000*

[5] *Eldar, Y., & Kutyniok, G. (Eds.). (2012). Compressed Sensing: Theory and Applications. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511794308*

[6] *Norbert Goertz, TU Wien Compressive Single-Pixel Imaging SuS2 06/2018*

[7] *Boyd, S. & Vandenberghe, L. (2004) Convex Optimization. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511804441*

[8] *http://www.junlulocky.com/blog/L1vsL2*

[9] *D. Donoho & X. Huo, "Uncertainty principle and ideal atomic decomposition," IEEE Trans on Information Theory, vol. 47, pp. 2845–2862, 2001.*

[10] *Y. Tsaig & D. Donoho, "Breakdown of equivalence between the minimal l1-norm solution and the sparsest solution," Signal Processing, vol. 86, no. 3, pp. 533–548, 2006.*

[11] *D. L. Donoho & M. Elad, "Optimally sparse representation in general (nonorthogonal)dictionaries via L1 minimization," Proceedings of the National Academy of Sciences, vol.100, no. 5, pp. 2197–2202, 2003.*

[12] *J.-J. Fuchs, "On sparse representations in arbitrary redundant bases," Information Theory, IEEE Transactions on, vol. 50, no. 6, pp. 1341–1344, 2004.*

[13] *http://www.simonlucey.com/lasso-using-admm/*

[14] *S. Boyd, N. Parikh, E. Chu, B. Peleato, & J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning, 3(1):1–122, 2011.*

[15] *S. G. Mallat & Z. Zhang. Matching pursuits with time-frequency dictionaries. IEEE Trans. Signal Proc., 41:3397–3415, 1993.*

[16] *Y. C. Pati, R. Rezaiifar, & P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In Proc. of the 27th Asilomar Conference on Signals, Systems and Computers, 1:40-44, 1993.*

[17] *D. Donoho, Y. Tsaig, I. Drori, & J.-L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," IEEE Transactions on Information Theory, vol. 58, no. 2, pp. 1094–1121, Feb 2012.*

[18] *D. Needell & . A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," Applied and Computational Harmonic Analysis, vol. 26, no. 3, pp. 301–321, 2009.*

[19] *A. Maleki, "Approximate message passing algorithms for compressed sensing," Ph.D. dissertation, Stanford University, Stanford, CA, USA, Jan. 2011*

[20] *Compressed Sensing and the Single-Pixel Camera Norbert Goertz, TU Wien*

[21] *Daubechies, M. Defrise, C. De Mol, 2003, 2004 https://doi.org/10.1002/cpa.20042*

[22] *Thomas Blumensath & Mike E. Davies 2009 https://doi.org/10.1016/j.acha.2009.04.002*

[23] *Andrea Montanari. Graphical models concepts in compressed sensing, https://arxiv.org/abs/1011.4328*

[24] *Bayati, M., Montanari, A. 2011. The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing. IEEE Transactions on Information Theory, 57(2), 764–785.*

[25] *Donoho, D.L., Maleki, A., Montanari, A. 2009. Message-passing algorithms for compressed sensing. Proceedings of the National Academy of Sciences, 106(45), 18914–18919.*

[26] *Donoho, D.L., Maleki, A., Montanari, A. 2011. How to Design Message Passing Algorithms for Compressed Sensing. unpublished draft, accessed 30 Dec. 2013, Feb.*

[27] *C. A. Metzler, A. Maleki, & R. G. Baraniuk, "From denoisign to compressed sensing," arXiv:1406.4175v3 [cs.IT], July 2014.*

[28] *D. Donoho, A. Maleki, & A. Montanari. Message passing algorithms for compressed sensing: I. motivation and construction. In IEEE Information Theory Workshop, 2010*

[29] *Wang, Z., & Bovik, A.C. (2004). Foveated Image and Video Coding.*

[30] *Z. Wang, A. C. Bovik & H. R. Sheikh, "Structural similarity based image quality assessment," in Digital Video Image Quality and Perceptual Coding (H. R. Wu, & K. R. Rao, eds.), Marcel Dekker Series in Signal Processing and Communications, Nov. 2005.*

[31] *Philip Kortum & Wilson Geisler Implementation of a foveated image coding system for image bandwidth reduction University of Texas Center for Vision and Image Sciences. Austin, Texas 78712 SPIE Proceedings, 2657, 350-360, 1996*

[32] *David B. Phillips,1* Ming-Jie Sun,1,2* Jonathan M. Taylor,1 Matthew P. Edgar,1 Stephen M. Barnett,1 Graham M. Gibson,1 Miles J. Padgett1 Adaptive foveated single-pixel imaging with dynamic supersampling*

[33] *Marco F. Duarte, Mark A. Davenport, Dharmpal Takhar, Jason N. Laska Ting Sun, Kevin F. Kelly, Richard G. Baraniuk Rice University Single-Pixel Imaging via Compressive Sampling*

[34] *R. Larcom & T. Coffman, Foveated image formation through compressive sensing, Proc. of Southwest Symp. Image Anal. Interp., (2010), 145-148. doi: 10.1109/SSIAI.2010.5483896*

[35] *Iulian B. Ciocoiu Circuits Syst Signal Process (2015) 34:1001–1015 DOI 10.1007/s00034-014-9878-2 © Springer Science+Business Media New York 2014*

[36] *Zhou Wang & Alan C. Bovik Foveated Image and Video Coding*