EXPLORATION OF COMPRESSED SENSING FOR SATELLITE CHARACTERIZATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Daigo Kobayashi

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Aeronautics and Astronautics

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Professor. Carolin E. Frueh, Ph.D., Committee Chair
School of Aeronautics and Astronautics
Professor. David A. Spencer, Ph.D., Committee Member
School of Aeronautics and Astronautics
Professor. Kathleen C. Howell, Ph.D., Committee Member
School of Aeronautics and Astronautics

Approved by:

Professor. Gregory Blaisdell
Head of the Graduate Program

Dedicated to my mother and father.

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Carolin Frueh for all the guidance she has given me since I joined her group. I would not be able to finish my work without her help.

I am also deeply thankful to my colleagues. They always stay in an office together and have supported me both personally and professionally. I also want to thank my friends at Purdue who spared me their valuable time.

Last but not the least, I am grateful to the School of Aeronautics and Astronautics and Ito Foundation for International Education Exchange for their exceptional facilities and financial support during my time at Purdue.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

SYMBOLS

$\bar{N}$	Mesh normal vector
$\bar{V}$	Satellite to Observer vector
$\bar{S}$	Satellite to Sun vector
$\bar{r}_{obs}$	Earth center to Observer vector
a_{Sun}	Distance between Earth and Sun
r_{Sun}	Solar radius
$\bar{r}_{Sun}$	Geocenter to Sun vector
$\bar{r}_{sat}$	Geocenter to Satellite vector
r_{topo}	Distance between Observer and Satellite
C_s	Specular reflection parameter
C_d	Diffuse reflection parameter
I_0	Solar intensity
$\bar{y}$	Measurement in compressed sensing
$\bar{x}$	Vectorized image
$\bar{\gamma}$	Sparse representation vector of image
$\bar{\rho}$	Mesh-wise light intensity vector
n_{pixel}	Number of pixels in image
$\mathbf{\Phi}$	Sensing matrix
$\mathbf{\Psi}$	Sparsifying basis matrix
$\mathbf{D}$	Dictionary
$\mathbf{Y}$	Measurement matrix whose columns are $\bar{y}$
$\mathbf{X}$	Image matrix whose columns are $\bar{x}$
$\mathbf{Y}_{\text{ref}}$	Measurement matrix of reference satellite
$\mathbf{X}_{\text{ref}}$	Image matrix of reference satellite

ABBREVIATIONS

SSA	Space situational awareness
BRDF	Bidirectional reflectance distribution function
DCT	Discrete cosine transform
RIP	Restricted Isometry Property
CS	Compressed sensing
OMP	Orthogonal matching pursuit
KSVD	K-Singular Value Decomposition algorithm
ADMM	Alternating direction method of multipliers
PSNR	Peak signal to noise ratio
PWM	Pulse width modulation
LC	Light curve
DDA	Digital differential analyzer
DMD	Digital micro-mirror device

ABSTRACT

Kobayashi, Daigo M.S., Purdue University, May 2020. Exploration of Compressed Sensing for Satellite Characterization. Major Professor: Carolin Frueh Professor.

This research introduces a satellite characterization method based on its light curve by utilizing and adapting the methodology of compressed sensing. Compressed sensing is a mathematical theory, which is established in signal compression and which has recently been applied to an image reconstruction by single-pixel camera observation. In this thesis, compressed sensing in the use of single-pixel camera observations is compared with a satellite characterization via non-resolved light curves. The assumptions, limitations, and significant differences in utilizing compressed sensing for satellite characterization are discussed in detail. Assuming a reference observation can be used to estimate the so-called sensing matrix, compressed sensing enables to approximately reconstruct resolved satellite images revealing details about the specific satellite that has been observed based solely on non-resolved light curves. This has been shown explicitly in simulations. This result implies the great potential of compressed sensing in characterizing space objects that are so far away that traditional resolved imaging is not possible.

1. INTRODUCTION

The space around the Earth is occupied by 900,000 man-made objects which are 1cm or larger [1]. Most of them are fragments of unused spacecraft, generated by random collisions between objects. These fragments can be a new source of collisions and hence the population growth of small debris is said to be exponential even without any launches of spacecraft [2]. In order to protect satellites from such a threat, it is necessary to detect new objects, track the detected objects, and characterize the tracked objects to predict their locations at any given time. These are the core ideas of space situational awareness (SSA). As a part of the SSA, this research focuses on the characterization of unknown space objects around Earth.

The main goal of the characterization is to determine the size, shape and attitude of unknown space objects. There exist some methods to tackle this problem by using an imaging approach. One of the most acclaimed examples is the FGAN Tracking and Imaging Radar (TIRA) [3], which consists of a 34-m parabolic antenna, an L-band traking radar and a high-resolution Ku-band imaging radar. This Ku-band radar was used to investigate the cause of the malfunction of the Advance Earth Observation Satellite (ADEOS) in 1997 and researchers succeeded in finding dislocated solar panels based on the radar image as shown in Figure 1.1 (b). Douglas et al. [4] suggested an imaging method for space objects under strong atmospheric turbulence. They modified multi-frame blind deconvolution (MFBD) algorithm for aperture-diverse data and used a bootstrap approach in post-processing to obtain high-resolution images of the Hubble Space Telescope (HST) in simulations. The simulations are implemented by assuming optical observations with and without a wave front sensor. These methods deal with 10-m size satellites in Low Earth orbit (LEO). Low Earth orbits have an altitude up to 1000 km above the Earth surface.

However, in fact, most earth-orbiting objects are too small or remote to be imaged by even the most state-of-the-art ground-based instrument.

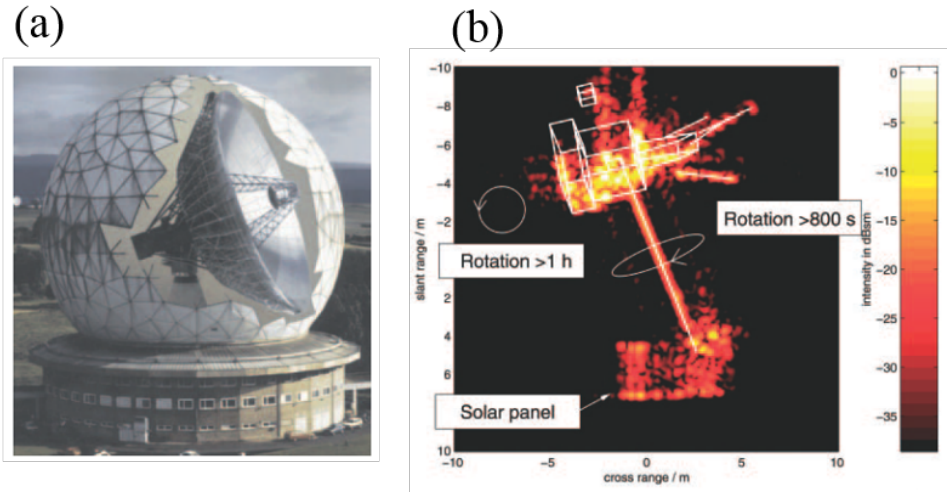


Fig. 1.1.: (a) The TIRA facility, (b) Radar image of ADEOS used in damage analysis [3], p.130, p.134

On the other hand, light curve-based method can be applied to objects in higher orbits including geosynchronous orbits with an altitude of 35786 km. The basic idea is to estimate shape or orientation of an object based on its light curve, which is a time history of intensity of light reflected on an object. This approach has been used to characterize celestial objects. M.Kaasalainen ad J.Torppa [5] [6] succeeded in obtaining a three-dimensional shape of asteroids including a nonconvex object. Later, light curve-based approach has also been used for artificial satellites. However, characterization of man-made objects requires additional efforts because of their complex surface properties, unstable attitudes, and highly concave shapes. Calef et al. [7] utilized thermal emissions and light curve to recover the three-dimensional shape of an object, assuming its orientation with respect to the observer is known and the object is convex. Linares et al. [8] used angles data and light curve to estimate a most probable shape of an object by using Unscented Kalman Filter. Linares and

Crassidis [9] obtained shape and surface parameters of a space object by Bayesian inversion approach. Recently, Furfaro et al. [10] used a deep learning method to classify the shape of space objects into four categories. Fan, Friedman and Frueh [11] implemented the light curve inversion considering a fact that real observations are greatly affected by noise and the conditions to obtain sufficient data for inversion in realistic settings.

Although much improvement has been made for the light curve inversion problem, it is still challenging to estimate attitude, shape and surface parameter simultaneously. Even when concentrating on the shape inversion problem alone, the problem is an ill-posed problem, which often has multiple solutions and ambiguities. Therefore, this research proposes an imaging method which reconstructs a resolved image of observed space objects based solely on the non-resolved light curve data. The method to be used is based on the so-called compressed sensing. Compressed sensing theory is capable of reconstructing an image that would be accurate enough to help estimate shape and surface parameters as engineers did with TIRA. Moreover, since light curves are available from almost all near-Earth objects, this method would also be applicable to geosynchronous objects, which cannot be imaged by previous imaging approaches.

Compressed sensing (CS) is a novel signal compression theory, which is capable of recovering compressible unknown signals from small number of random measurements even if they are inaccurate and incomplete. It compresses a signal as a linear measurement by using a so-called sensing matrix and reconstructs it by solving an optimization problem. The number of measurements required by CS is far fewer than those required by the traditional Shannon/Nyquist sampling theorem. Thus, it has given a paradigm shift in signal processing field. In the field of aerospace engineering, there have been some applications to enjoy this benefit. Aguilera et al. [12] [13] used CS in a synthetic aperture radar (SAR) tomography to improve the quality of reconstructions. Daponte et al. [14] used CS framework to design radio frequency (RF) sensors for localization and tracking of non-cooperative RF emitters. In re-

mote sensing, CS led to a new onboard instruments called MPST and SPMT camera, which need less storage space and less power consumption than classical CCD cameras [15] [16]. It is also possible to apply CS to obtain clear pictures from incomplete measurements [17]. However, there are no examples of CS applications to a light curve measurement.

Under some assumptions, a light curve of an object can be regarded as an object image compressed by atmospheric noise, which is considered as a pseudo-sensing matrix in CS framework. However, the more significant difference from the CS is that there is no information available about this pseudo-sensing matrix. Thus, the main interest of this research is how to estimate the original satellite image in the absence of pseudo-sensing matrix. One possible approach is to estimate a sensing matrix and a correct image at the same time by dictionary learning technique.

Dictionary learning is a learning method which has drawn a huge attention in signal processing community in the past ten years. It was first proposed by Olshausen and Field [18] in the context of studying brain cells. The basic idea is to find a common basis matrix to sparsely represent millions of images. This matrix is called a dictionary. The learned dictionary helps remove image noise [19], detect image edges [20], recognize image patterns [21], compress images [22] [23] or obtain super resolution [24] in a CS framework. Although these goals appear very different from the main interest of this research, there is a significant mathematical similarity between the estimation of the sensing matrix and the dictionary learning problem.

Another possible approach to estimate a sensing matrix and a resolved satellite image is to utilize a previous or simultaneous observation of a known satellite. This approach is similar to the technique called adaptive optics [25]. In astronomical observations, a star is obscured by unwanted wavefront distortions due to atmospheric turbulence. Adaptive optics technique removes the effect of the distortions by observing a reference star or an artificial laser-generated reference star [26]. This idea is also feasible for estimating a sensing matrix.

The organization of this thesis is as follows. In Chapter 2, light curve and rendering technique are introduced. The purpose of this chapter is to present the basic concept of light curve along with rendering technique. Using this rendering technique, satellite synthetic images are generated in Chapter 5. In Chapter 3, the method of compressed sensing and one specific application, a single-pixel camera are introduced. The concept of this single-pixel camera is important to understand the similarity between the compressed sensing and a light curve measurement. In Chapter 4, the method of dictionary learning is introduced along with one of its applications, denoising. The dictionary learning algorithm and the denoising algorithm is used in Chapter 5. In Chapter 5, the adaptation of the theory to the actual imaging of a satellite is shown via simulations. First, a simple problem with the knowledge of a sensing matrix is investigated through two simulations. The difference between these two simulations is the way of modeling a light curve. The first simulation uses a realistic light curve model while the second simulation uses a simpler adapted light curve model. In the subsequent section, dictionary learning approach is used to estimate a satellite image in the absence of a sensing matrix under two different assumptions. In the first simulation, a patch-based method is used to simplify the problem. In the second simulation, the same simulation is implemented without the patch-based method. Finally, in the last section, additional assumption is made to estimate a satellite image. A sensing matrix is still assumed to be unknown but a reference light curve and corresponding satellite images are assumed to be known.

2. LIGHT CURVE AND RENDERING

In this chapter, a method for a light curve simulation is introduced, followed by a result of light curve simulation of a GEO satellite. Subsequently, an overview of rendering technique is introduced. The light curve simulation and rendering share a common ground, and light curve can be regarded as an unresolved image of an object.

2.1 Light Curve Simulation

A light curve is defined as a series of brightness measurements of an observed object [27]. It is often plotted as magnitude versus time or phase as shown in Figure 2.1. In this case, the vertical axis shows the magnitude of an asteroid and the horizontal axis shows a fraction of the assumed period of the asteroid.

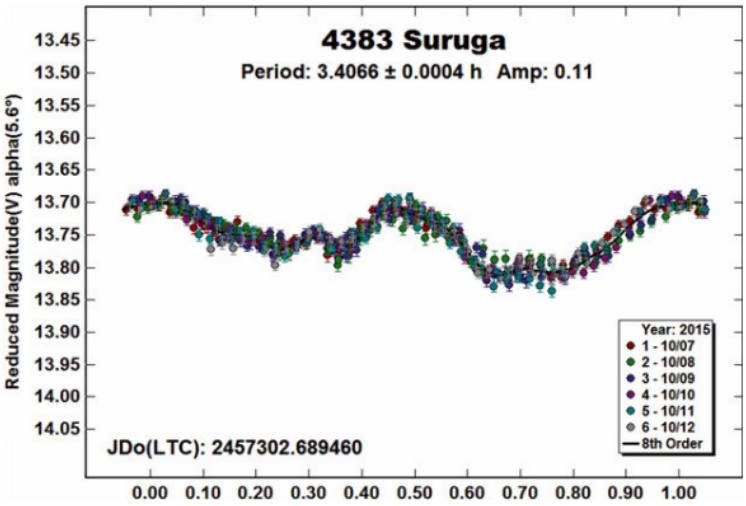


Fig. 2.1.: Light curve plot of 4383 Suruga [27], p.3

The magnitude is a measure of brightness, where the smaller magnitude means the brighter object. It is often defined relative to the brightness of the Sun. It is formulated as follows:

$$mag(t) = mag_{\text{Sun}} - 2.5 \log_{10} \left\{ \frac{I(t)}{I_0} \right\} \tag{2.1}$$

where mag_{Sun} is a magnitude of the Sun ($mag_{\text{Sun}} = -26.74$), $I(t)$ is the intensity of light from the object which is a function of time t, and I_0 is the intensity of the Solar intensity.

The intensity of the light curve is observed by collecting the light which is reflected off the surface of the space object and received by the observer. Therefore, it depends on the surface properties, such as shape and materials, and attitude of the object. For example, the satellite covered with aluminum (Al) tends to reflect more intense light compared to the one covered with silicon carbide (Sic) which absorbs most incoming light [28]. Moreover, the intensity also depends on astrophysical geometry between the object, the observer and the Sun since the intensity is determined by direction of incoming and outgoing light fluxes on the object.

The goal in the light curve simulation is to compute the intensity $I(t)$, which is a summation of the intensity of the light reflected from all the illuminated surfaces of the object in the direction of the observer. Figure 2.2 shows the flowchart of this simulation scheme.

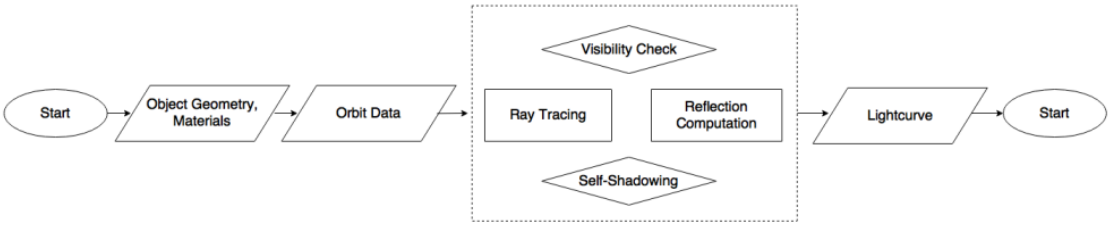
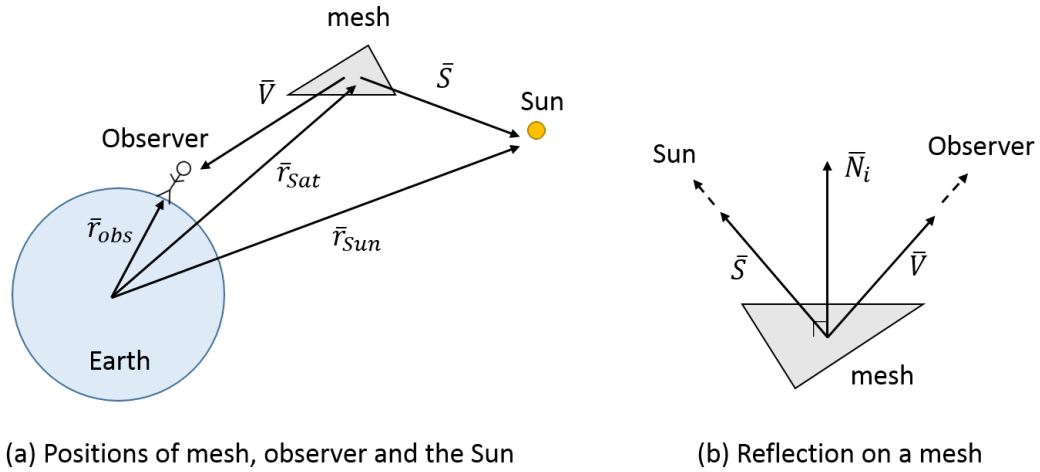


Fig. 2.2.: Simulation Flowchart [28]



(a) Positions of mesh, observer and the Sun

(b) Reflection on a mesh

Fig. 2.3.: Reflection on one mesh

However, before discussing a light reflection on a satellite, it is prudent to start with a reflection model of one single flat facet. Here this facet is called a mesh and its light intensity is computed by using BRDF model. Figure 2.3 shows the corresponding vectors. Its intensity I_i is computed with respect to the solar intensity [29]:

$$I_i = I_0 \cdot \frac{A_i}{(r_{\text{topo}})^2} \cdot p_i \tag{2.2}$$

where I_0 is the solar intensity at 1AU which is about $1365 W/m^2$, A_i is the area of the mesh and r_{topo} is the topocentric distance to the satellite. The term p_i is called a phase function which is a ratio between the incoming light flux and outgoing light flux on the mesh. This phase function consists of two terms: Lambertian reflection term $p_{\text{lamb, i}}$ and specular reflection $p_{\text{spec,i}}$:

$$p_i = p_{\text{lamb, i}} + p_{\text{spec,i}} \tag{2.3}$$

Note that the absorption term is neglected assuming the opaque surface. First, the Lambertian reflection term is computed as:

$$p_{\text{lamb, i}} = \frac{C_d}{\pi} \, (\bar{V} \cdot \bar{N}_i)(\bar{S} \cdot \bar{N}_i) \tag{2.4}$$

where C_d is the diffuse reflection parameter, which is the amount of light scattered equally in all directions. $\bar{V}$ is the-object-to-the-observer vector, $\bar{N}_i$ is the mesh normal vector and $\bar{S}$ is the-object-to-the-Sun vector. On the other hand, the specular reflection is a mirror-like reflection which is concentrated on a particular direction:

$$p_{\text{spec,i}} = \frac{\tau_i \; C_s \; (r_{Sun})^2}{a_{Sun}^2} \tag{2.5}$$

$$\tau_i = \begin{cases} 1 & \text{if } \; |\theta_{incoming} - \theta_{outgoing}| \leq 0.5^\circ \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where C_s is the specular reflection parameter, r_{Sun} is the solar radius and a_{Sun} is the distance from the object to the Sun. The coefficient τ_i is either 0 or 1 depending on the angle between the incoming and outgoing light flux. It becomes 1 when the angle is within half a degree as shown in Eq.(2.6).

Therefore, the light intensity of the flat mesh is computed by substituting Eq.(2.3), Eq.(2.4) and Eq.(2.5) into Eq.(2.2):

$$I_i = I_0 \cdot \frac{A_i}{(r_{\text{topo}})^2} \left\{ C_d \; (\bar{V} \cdot \bar{N}_i)(\bar{S} \cdot \bar{N}_i) + \frac{\pi \tau_i \; C_s \; (r_{Sun}(t))^2}{a_{Sun}^2} \right\} \tag{2.7}$$

Based on this reflection model of one mesh, the light curve of a complete satellite is computed. First, a 3D satellite geometry is constructed by using Solidworks. The geometry is modeled as a collection of meshes which are small enough to approximate curved surface. There are some kinds of meshes depending on the applications. In this research, triangle meshes are chosen since it is a polygon with the smallest number of edges and hence it can most accurately model a curved surface of parabolic antenna of a satellite, for example. The 3D satellite geometry is then characterized using Meshlab. At this stage, directions of normal vectors of each mesh are verified. Moreover, Lambertian reflection coefficient and specular reflection coefficient of each mesh are specified depending on its material.

An example of the 3D satellite model is shown in Figure 2.4. Finer meshes help to get more accurate light curve especially when considering self-shadowing and observer-shadowing. Both these shadowing effects are considered by solving an intersection

problem between a mesh and a light flux. A larger mesh is more likely to have the intersection with the light flux even if some parts of the mesh is actually not in a shadow. Therefore, the finer meshes are preferable in light curve simulation. This is in contrast to computer science, where a flat surface would be represented with fewest possible triangles.

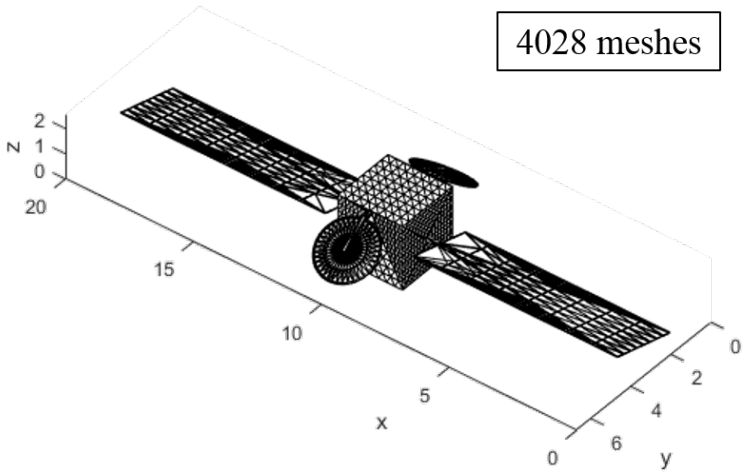


Fig. 2.4.: 3D satellite model made of triangle meshes

Second, orbit and attitude of the satellite are specified. The key information is the direction of each mesh with respect to the Sun and the observer.

Third, light intensity of each mesh is calculated. Before calculating the values, the visibility of each mesh is checked based on three conditions. The first condition is an observer's local horizon. A satellite has to be above the observer's local horizon:

$$-\bar{V} \cdot \bar{r}_{obs} > 0 \tag{2.8}$$

where $\bar{V}$ is an object-to-observer vector and $\bar{r}_{obs}$ is a position vector of the observer in geocentric frame. If a satellite does not meet this condition at a specific time, its light curve is not observable. The second condition is a mesh's local horizon. If the Sun is "behind" a mesh, the light does not reach to the mesh, so it is not observable.

Similarly, if the observer is "behind" a mesh, the mesh is not observable. In other words, the observer and the Sun has to be above a mesh's local horizon:

$$\bar{N}_i \cdot \bar{V} > 0 \quad \text{and} \quad \bar{N}_i \cdot \{\bar{r}_{Sun} - \bar{r}_{sat}\} > 0 \tag{2.9}$$

where $\bar{N}_i$ is the i^{th} mesh normal. $\bar{r}_{Sun}$ is the position vector of the Sun and $\bar{r}_{sat}$ is the position vector of the satellite in geocentric coordinates. The third conditions are self-shadowing and observer-shadowing. A mesh must not be hidden by other parts of the satellite. There are two such cases as shown in Figure 2.5. In (a), a light flux coming into mesh 2 is obstructed by mesh 1, which is referred to as self-shadowing. In (b), a light flux coming out from mesh 2 is obstructed by mesh 1, which is referred to as observer-shadowing.

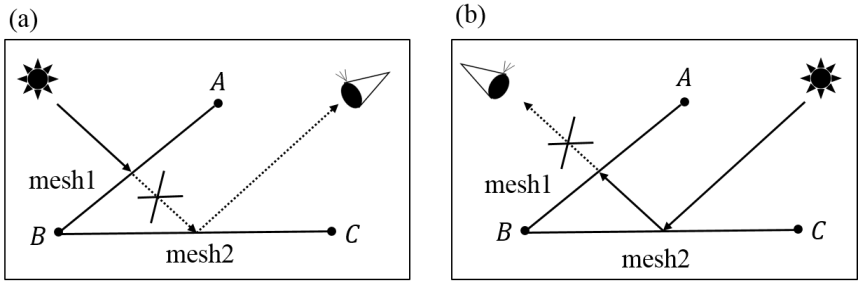


Fig. 2.5.: Self-shadowing and observer shadowing

Therefore, the light flux coming in and out on the mesh must not have any intersections with any other meshes. In order to check this intersection problem, Moller-Trumbore intersection algorithm [30] is used.

The Moller-Trumbore intersection algorithm determines whether a light flux coming from point C and the triangle $P_1P_2P_3$ have the intersection. Eq.(2.10) represents an arbitrary point on the light flux from point C, and Eq.(2.11) represents a collection of all the points in the triangle $P_1P_2P_3$.

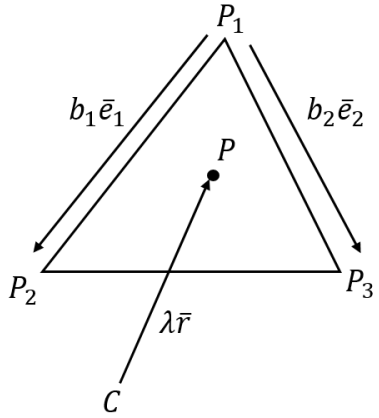


Fig. 2.6.: Moller-Trumbore intersection

$$P = C + \lambda\bar{r} \quad \text{where } \lambda \leq 0 \tag{2.10}$$

$$\begin{aligned} P &= P_1 + b_1(P_2 - P_1) + b_2(P_3 - P_1) \\ &= P_1 + b_1\bar{e_1} + b_2\bar{e_2} \quad \text{where } 0 \leq b_1 \leq 1,\ 0 \leq b_2 \leq 1,\ b_1 + b_2 \leq 1 \end{aligned} \tag{2.11}$$

Substituting Eq.(2.10) into Eq.(2.11) and rearranging the equation gives:

$$\begin{bmatrix} -\bar{L} & \bar{e}_1 & \bar{e}_2 \end{bmatrix} \begin{bmatrix} \lambda \\ b_1 \\ b_2 \end{bmatrix} = C - P_1 \tag{2.12}$$

Therefore, if the solution of Eq.(2.12) satisfies the conditions:

$$\lambda \leq 0,\ 0 \leq b_1 \leq 1,\ 0 \leq b_2 \leq 1,\ b_1 + b_2 \leq 1 \tag{2.13}$$

then the light flux has an intersection with the triangle $P_1P_2P_3$. This algorithm is used to check both a vector from the Sun to a mesh and a vector from the observer to a mesh to see whether it has an intersection with any other meshes.

Finally, a light curve is obtained. The meshes satisfying all the three conditions are regarded as visible. For each of these visible meshes, the light intensity is computed

by Eq.(2.2). The summation of the intensities of all the visible meshes is the value of the light curve $I(t)$ at specific time:

$$I(t) = I_0 \sum_{i=1}^{n} \frac{A_i}{\pi (r_{topo}(t))^2} \left\{ C_d \ (\bar{V} \cdot \bar{N}_i)(\bar{S} \cdot \bar{N}_i) + \frac{\pi \tau_i \ C_s \ (r_{Sun}(t))^2}{a_{Sun}^2} \right\} \tag{2.14}$$

By computing this intensity in a discrete time frame, a light curve is obtained.

Turning now to an example of a light curve, it is simulated by the following procedure. A model in Fig 2.8 is used as a 3D satellite model. This model consists of 4028 meshes and all the meshes are assumed to have the same surface property: $C_d = 0.2,\ C_s = 0.8$ assuming that the satellite is made of aluminum (Al), which is one of the most common materials for spacecraft. The reflection property highly depends on finish but its specular reflection coefficient is often assumed to be around 0.8-0.9 [28]. Its orbit is computed based on the TLE data of Astra 1KR satellite obtained from Space-track organization website [31]. It is propagated for its orbital period, 1437 minutes by SGP4 propagator. Figure 2.7 shows the orbit of the satellite in this simulation. The orange arrow shows the direction of the Sun and the red dot shows the satellite position at the beginning of this simulation. The motion of the Sun is considered but its direction hardly changes in this short simulation time.

The orientation of the satellite also needs to be defined. Active satellites are traditionally either spin-stabilized or three-axis stabilized. Here, a three-axis stabilized satellite is assumed. Since the parabolic antennas always need to be oriented toward the Earth, one facet of the satellite body (red arrow in Figure 2.8) is fixed to direction of the geocenter, and another facet (blue arrow in Figure 2.8) is fixed so that it is always perpendicular to its orbital plane. Therefore, the solar panels are not always oriented toward the Sun in this simplified simulation. The observer is located in Berlin and the observation starts at 0 UTC on 1-April-2020. All the parameters used in this simulation are shown in Table 2.1.

Table 2.1.: Parameters in the light curve simulation

Name		Value / Description
Observer's location		51.17 N, 10.45 E
Observation duration		0:00:00 am - 11:56:00 pm on 1-April-2020
Lambertian reflection coefficient	C_d	0.2
Specular reflection coefficient	C_s	0.8
Radius of Earth	R_{Earth}	6378.136 km
Radius of Sun	R_{Sun}	695990 km
Irradiance of the Sun at 1AU	I_0	1365 W/m2
Orbit		Astra 1KR (GEO)

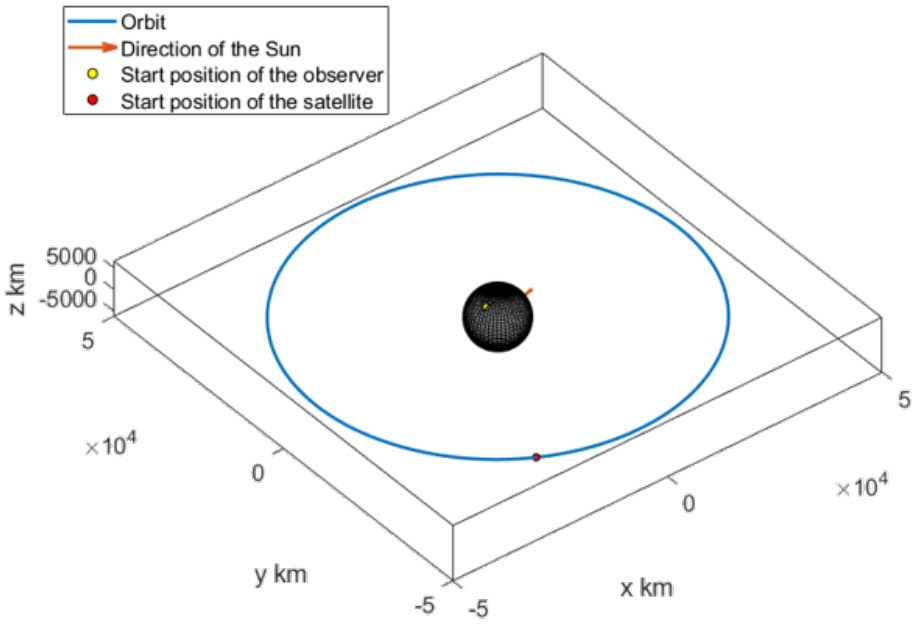


Fig. 2.7.: Satellite orbit in the light curve simulation

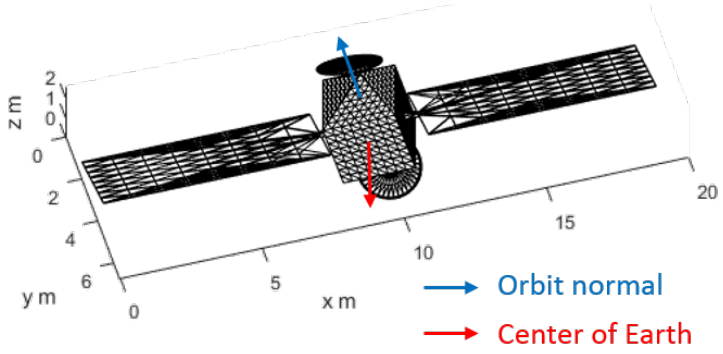


Fig. 2.8.: 3D satellite model used for simulation

The result is shown in Figure 2.9. First, the magnitude is around 2.6 but it rapidly increases. At $t = 178$ minutes, it decreases sharply but it again starts to increase. At $t = 125$ minutes, its change rate becomes slower. At around $t = 500$ minutes, the magnitude becomes the largest. At this time step, the satellite almost hides the Sun from the observer as shown in Figure 2.10 (a). After this, the magnitude gets smaller and at $t = 850$ minutes, the magnitude suddenly drops dramatically. At this time step, the Sun direction is almost perpendicular to the line of sight of the satellite as shown in Figure 2.10 (b), which means the sunlight is almost parallel to the solar panel normal. Therefore, before this step, the solar panel does not reflect sunlight to the observer at all but after this time step, the solar panel starts to reflect a light to the observer, which explains the sudden drop of the magnitude. Later, the magnitude gets the smallest around $t = 1250$ minutes when the solar panel normal is almost parallel to the sunlight as shown in Figure 2.10 (c).

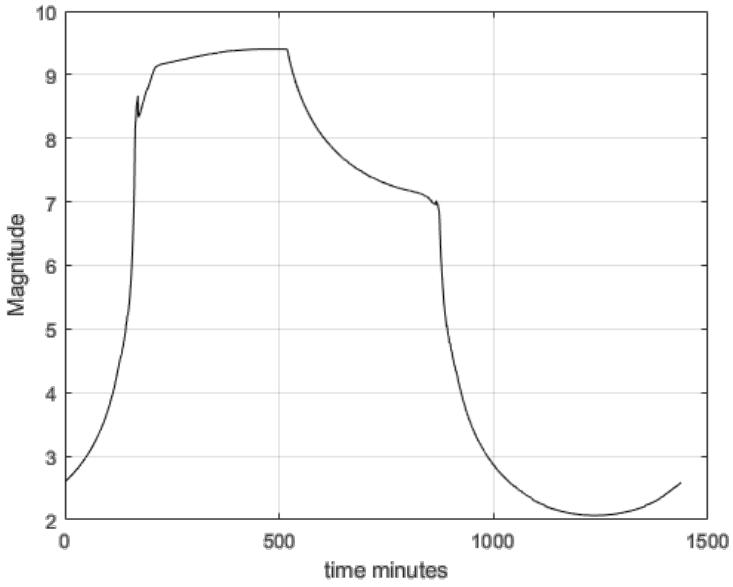


Fig. 2.9.: Simulated light curve

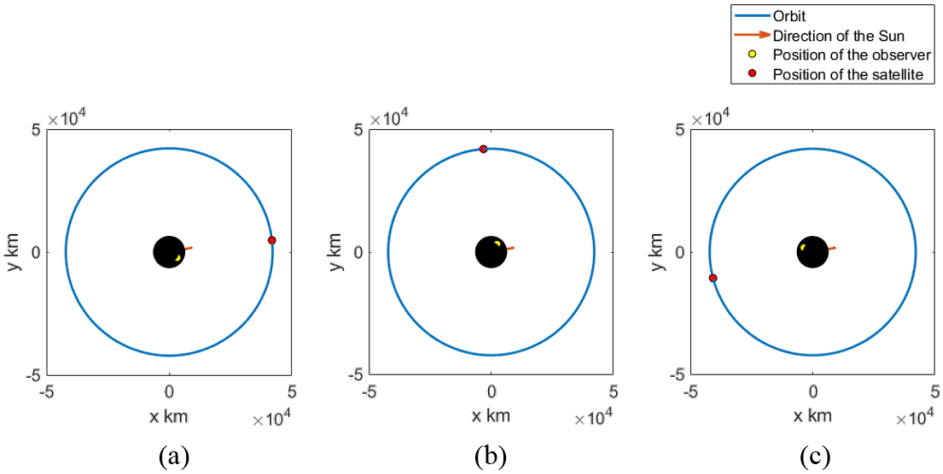


Fig. 2.10.: Position of satellite, observer and the Sun at each time step: (a) t = 500 [min], (b) t = 850 [min], (c) t = 1250 [min]

2.2 Synthetic Image Generation by Rendering Technique

This section introduces a scheme for synthetic image generation using rendering technique. Rendering technique is used to visualize a 3D object in the form of 2D image. In contrast to the light curves, a resolved image is simulated. The basic idea is to set an imaginary image plane consisting of grids, which are referred to as pixels, in front of the observer. As shown in Figure 2.11, a light from the light source reflects on the surface of the object, passes through one of the pixels in the image plane, and reaches to an observer. This observed light intensity is assigned to the pixel it has come through. This process is repeated for all the light fluxes until all the pixel values are specified. In the following, these steps will be introduced in details [32].

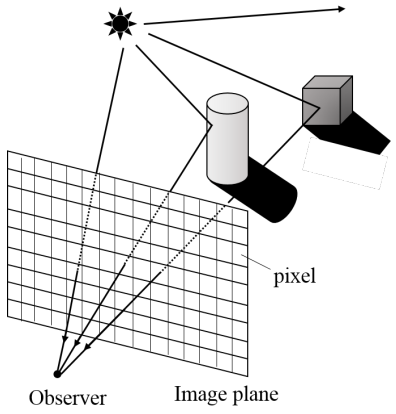


Fig. 2.11.: Basic idea of the rendering technique

The first step is to consider a reflection of the meshes. In this step, the brightness of each mesh of a 3D model is computed. This computation is exactly the same with the light curve simulation introduced earlier. In a computer graphics field, there are some options for the reflection models to obtain more realistic appearances. However, from a physical perspective, a specular and Lambertian reflection model is chosen in

this research. This also ensures consistency with the light curve simulation. The brightness I_i of the i^{th} mesh is:

$$I_i = I_0 \cdot \frac{A_i}{\pi(r_{topo})^2} \left\{ C_d \ (\bar{V} \cdot \bar{N}_i)(\bar{S} \cdot \bar{N}_i) + \frac{\pi \tau_i \ C_s \ (r_{Sun})^2}{a_{Sun}^2} \right\} \tag{2.15}$$

$$\tau_i = \begin{cases} 1 & \text{if } \ |\theta_{incoming} - \theta_{outgoing}| \leq 0.5^\circ \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

Note that A_i is the area of the mesh, r_{topo} is the distance between the observer and the object, $\bar{V}$ is the object-to-observer vector, $\bar{S}$ is the object to the Sun vector, $\bar{N}_i$ is the i^{th} mesh normal, r_{Sun} is the solar radius and a_{Sun} is the distance between the Sun and the Earth. θ_{incoming} and θ_{outgoing} are the angles of incoming and outgoing light flux with respect to the mesh normal. C_s is a specular reflection parameter and C_d is a diffuse reflection parameter. Assuming that the satellite is coated with aluminum, these parameters are set to be $C_s = 0.8, \ C_d = 0.2$ for all the meshes.

The next step is a projection of the 3D model onto a 2D plane. In this step, the geometry of each mesh on the image plane needs to be computed by orthogonal projection. First, a coordinate transformation matrix is obtained such that an object-to-viewer vector is parallel to a z-axis in a new Cartesian coordinate. Subsequently, all the coordinates of the vertices in the 3D model are transformed to this new coordinate. The z coordinates at this stage are called depths. This will be significant information in a hidden-surface removal procedure, so it is saved in a certain form. Finally, the z-coordinates of all the vertices are set to be zero. At this stage, the values of the coordinates correspond to the satellite dimension, so the values are need to be stretched to fit into an image. A satellite is considered to be circumscribed by a rectangle whose edges are parallel to x and y axis. The longer edge is stretched to the size of one edge of a square image. After this projection step, all the vertices have been projected regardless of their z-coordinates. This means, even a vertex or an edge which are supposed to be hidden are displayed in an image. This issue will be dealt with in a later procedure.

The third step is a rasterization. In general, an image is expressed on a grid. The cells of the grid is called pixels and each pixel color is determined by the pixel value. Therefore, the geometry of the satellite needs to be expressed as a collection of pixels. This approximation technique is called a rasterization. First, all the coordinate of the vertices obtained in the previous subsection are rounded to the nearest integers, since the position of all the pixels in an image is expressed in terms of integer coordinates. Since all the meshes are triangles consisting of these vertices, the next step is to rasterize arbitrary triangles.

All triangles are expressed as a collection of line segments. Therefore, it is important to know how to rasterize a line segment first. The simplest algorithm for line segments is known as a DDA (digital differential analyzer) algorithm [32]. Suppose that a line segment is defined by two points (x_1, y_1) and (x_2, y_2) as shown in Figure 2.12, then the slope is given by:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} \tag{2.17}$$

Note that the slope is positive and smaller than 1:

$$0 \leq m \leq 1 \tag{2.18}$$

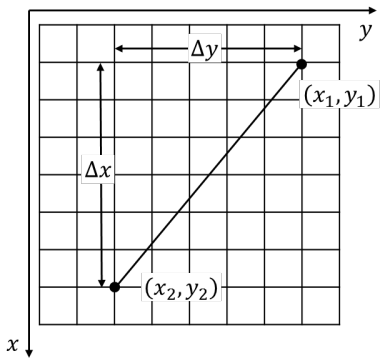
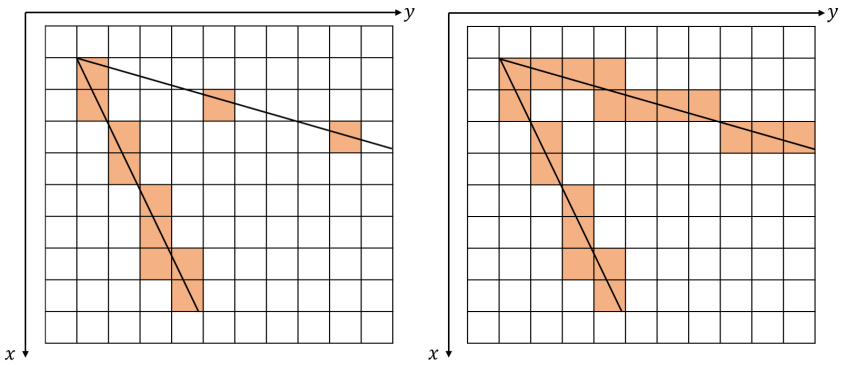


Fig. 2.12.: Line segment in image coordinate



(a) Pixels generated by high- and low-slope lines (b) Pixels generated by revised DDA algorithm

Fig. 2.13.: Pixels generated by DDA before and after revision

Under this assumption, the best y is computed for each x. In each iteration, x is increased by 1 and the corresponding y value is computed. Since m is a floating-point number, the result of $m\Delta x$ needs to be rounded.

$$y = y_1 + \text{round}(mx) \quad x = 1, 2, ..., \Delta x \tag{2.19}$$

However, if the slope is greater than 1, this algorithm fails to draw a correct line segment as shown in Figure 2.13 (a). This issue can be solved by swapping the roles of x and y, and applying the same algorithm. As a result, Figure 2.13 (b) is obtained.

Next, a procedure of rasterizing a triangle is discussed. The simplest algorithm is known as a scanline algorithm [32]. Consider a triangle shown in Figure 2.14 (a) with n lines drawn so that they have intersections with the triangle. These lines are called scanlines. The first scanline has two intersections P_{11} and P_{12} with two edges of the triangle. A group of pixels on the line segment $P_{11}P_{12}$ is called a span. The scanline is specified only by x value on the image plane. Once the scanline is specified, the intersections on the scanline can be specified by y value. Therefore, a bucket is created for each scanline to save these intersections as shown in Figure 2.14 (b). As the scan proceeds, the x values of intersections are saved in proper buckets.

Subsequently, this data structure is used to fill the pixels circumscribed by this triangle. Given the positions of two intersections, the span between the two inter-

sections is filled by using the DDA algorithm discussed earlier. This procedure is repeated for all the scanlines and a triangle image is obtained in the image plane.

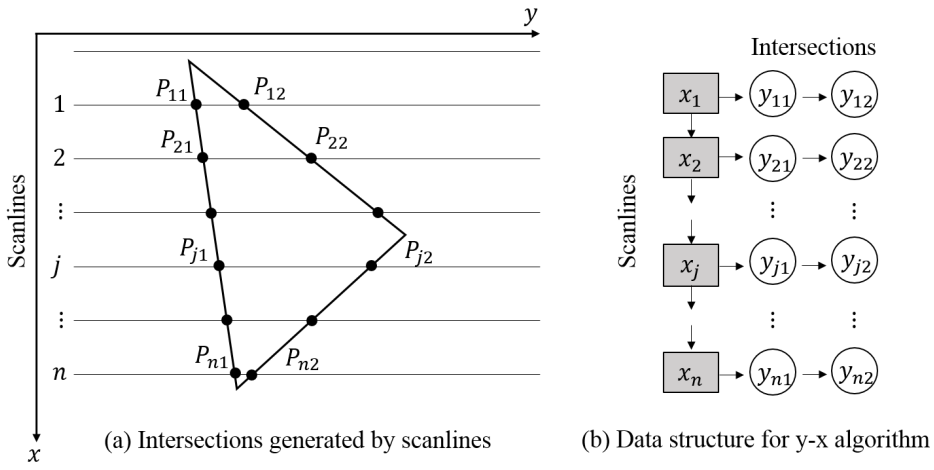


Fig. 2.14.: Basic idea of scanline algorithm

Next, pixel values are assigned to each pixel based on the brightness of each mesh obtained in the first step. There are mainly three ways for this so-called shading technique: flat shading, Gouraud shading, and Phong shading. The Gouraud shading and Phong shading determines pixel values so that there are some gradations even in one mesh. This helps objects in an image look more realistic. However, for simplicity, this research uses a flat shading, which assigns one solid color in a mesh. This also helps to create consistency with the light curve simulations. One pixel value in a mesh is obtained by simply dividing the brightness of a mesh by the total number of pixels contained in the mesh.

The final step is to remove extra pixels from an image, which is done concurrently with the rasterization. As can be seen in Figure 2.18, the projection of all the vertices onto an image plane even shows surfaces (meshes) that are supposed to be hidden behind. A part of the pixels in these meshes are needed to be removed from an image in a systematic way. The most widely used approach is the z-buffer algorithm.

The basic idea of the z-buffer algorithm is as follows. In Figure 2.15, a mesh A is partly hidden by a mesh B. Pixels in the overlap of A and B have two possible values, p_A and p_B, where p_A is a value obtained by brightness of A, and p_B is a value obtained by brightness of B. However, since $z_2 > z_1$, the value p_B is chosen. In general, when a pixel belongs to multiple meshes, a correct pixel value is determined based on a mesh with the largest z value (depth).

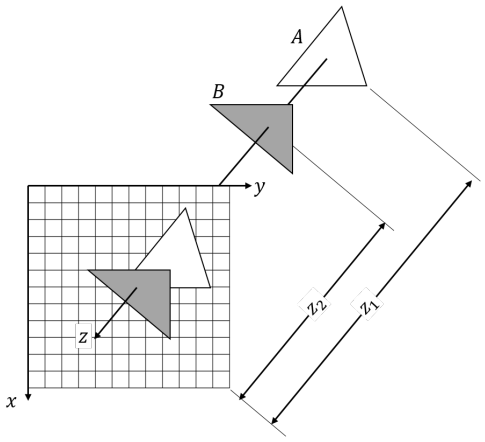


Fig. 2.15.: Z-buffer algorithm

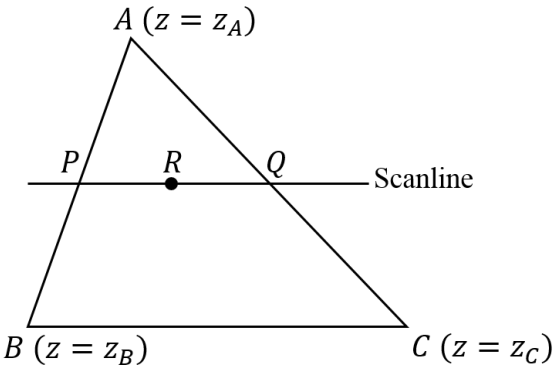


Fig. 2.16.: Bilinear interpolation to compute a depth of an arbitrary point R

Therefore, the rasterization procedure is slightly modified when considering depths. When computing the pixel value by DDA algorithm, the depth of the corresponding point is also computed by bilinear interpolation method. This technique also uses a scanline. For example, a depth of pixel R is computed as follows. This pixel is surrounded by a triangle mesh ABC in Figure 2.16. This pixel is on a scanline which has intersections P and Q with triangle ABC. Since the depths of pixels A and B are known, the depth of P can be obtained by linear interpolation of these two depths. The depth of Q can also be obtained in a similar way. Given the depths of P and Q, the depth of R can be obtained by linear interpolation of the depths of P and Q. This is the depth of the pixel R obtained by a triangle mesh ABC but not necessarily the true depth of the pixel R. This pixel may belong to other meshes which are closer to the observer. If it is the case, the mesh ABC is hidden by other mesh. Thus, this depth value is saved as a candidate solution. This computation is repeated for all the pixels on the visible meshes, and every time the depth of each pixel is calculated. If the depth value is larger than the candidate value, this new value is chosen as a candidate solution, and the DDA algorithm assigns a signal value to the pixel. If the depth value is smaller than the candidate value, the DDA algorithm does not do anything. Sweeping through all the meshes, we get the pixel values considering the right depth values.

By combining all these techniques: reflection, projection, rasterization, shading and hidden-surface removal, an image of a 3D satellite model is obtained.

Turning now to an example, a satellite image is constructed based on a 3D satellite model. Figure 2.17 shows the 3D satellite model. The number of meshes are 4028 and the number of vertices are 2016. The red arrow shows the object-to-Sun vector, $\bar{L}$, and the blue arrow shows the object-to-observer viewer vector, $\bar{V}$. The goal is to obtain an image when this satellite is seen from a direction of $\bar{V}$.

First, the coordinate of the satellite is transformed to a new coordinate so that $\bar{V}$ is parallel to a z axis in a new coordinate as shown in Figure 2.18 (a). Subsequently,

the coordinate is stretched so that it fits into an image frame of size 512×512, and Figure 2.18 (b) is obtained.

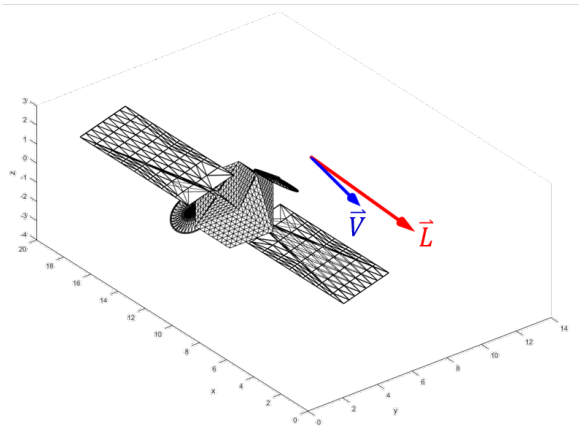


Fig. 2.17.: Example of a 3D model

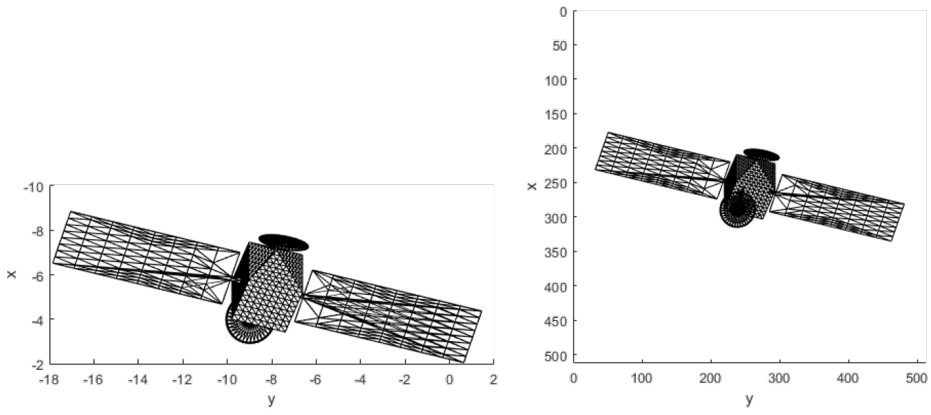


Fig. 2.18.: Example of coordinate transformation and projection

This is a projection of the 3D model onto the xy-plane (the image plane). Note that even a hidden mesh is visible in Figure 2.18 (b). This issue is taken care of by a hidden-surface removal procedure. Figure 2.19 shows a depth of all the pixels of

the satellite model computed by bilinear interpolation. The brighter pixel has larger z-coordinate, which means closer to the observer.

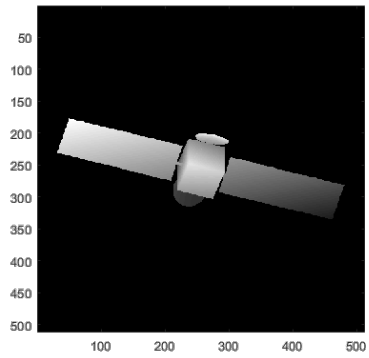


Fig. 2.19.: Depth information of the 3D model

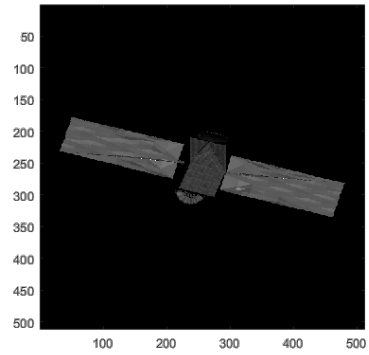


Fig. 2.20.: Example of a synthetic satellite image

The result is shown in Figure 2.20. The upper antenna dish is almost not illuminated, which makes sense since it is almost parallel to the incoming light as shown in Figure 2.17. Moreover, the solar panel looks brighter than the front panel of the satellite body since the angle between the solar panel normal and the light flux is smaller than that of the front panel and the light flux.

However, there are some errors in the image. The most remarkable error is a mosaic pattern in the image. Ideally, all the meshes in the solar panel, for example,

should be shown in the same color but it is not the case with this image. This is caused in a conversion from mesh intensity to pixel values. For example, even if the areas and normal vectors of two meshes are exactly the same, they are not necessarily expressed by the same number of pixels. As a result, the pixels in the two meshes have different values. There are some ways to solve this issue. One way would be to use a different method to assign a value to a pixel. In the current method, the pixel value assignment is done in a mesh-wise way which causes this issue. This issue may be solved if the assignment is done in a facet-wise way. If the meshes are regarded to be on the same facet, the pixel value assignment is done in the facet independently. Another way would be to make the meshes even smaller so that each mesh correspond to one pixel. However, the larger number of meshes makes the computation more expensive.

Another important point to note is that the sum of all the pixel values of this image is approximately the same with the intensity of the light curve of this satellite. This is because the BRDF model and flat shading are used in the reflection step to construct this image. This property will be important in the later chapters.

3. COMPRESSED SENSING

3.1 Compressed Sensing

3.1.1 Overview

Compressed sensing is a mathematical theory proposed by Candes, Romberg, Tao [33] [34] and Donoho [35] in 2006. It is also sometimes referred to as compressive sampling or compressed sampling. It has been widely used in signal compression method such as JPEG, JEPG2000, MPEG and MP3 standards.

Its advantage is often described by comparing it with a classical sampling theory in the Nyquist-Shannon framework. The classical sampling scheme consists of two steps. The first step is a sampling, in which infinite-length continuous-time signals are digitized by a set of uniformly spaced samples. The Nyquist-Shannon theory [36] states that the signal can be exactly recovered if the sampling rate is larger than a specific threshold, which is twice the value of Nyquist rate. However, this Nyquist rate is high, which makes the amount of samples (represented by a vector with N elements) too large to store in devices efficiently. This issue is addressed in the second step: compression. The data is represented in the most concise form in an allowable distortion range. The most popular compression technique is known as transform coding. A basis of the high-dimensional samples is found so that it can be represented with only k ($\ll N$) coefficients. This overall classical sampling scheme has a massive redundancy because it needs to sample large amounts of data even though most of them are discarded in the compression step.

The compressed sensing solves this issue by developing the concept of transform coding. A signal is sampled directly in a compressed form:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{3.1}$$

where $\bar{x} \in \mathbb{R}^N$ is a signal, $\bar{y} \in \mathbb{R}^m (m < N)$ is a measurement, and $\mathbf{\Phi} \in \mathbb{R}^{m \times N}$ is a so-called sensing matrix. The sensing matrix compresses the signal by mapping a high dimensional vector into a lower dimensional vector. This enables us to sample a signal in a sampling rate which is much lower than that of the classical sampling scheme. If the signal $\bar{x}$ is sparse or compressible, which is often the case, and if the sensing matrix $\mathbf{\Phi}$ satisfies a certain condition, the signal $\bar{x}$ is guaranteed to be recovered from the measurement $\bar{y}$. The background theory is discussed in more details.

Signal models

For the success of the compressed sensing, a signal needs to be either sparse or compressible. A sparse signal is a vector with only k nonzero elements. In other words, the L_0 norm of $\bar{x}$, or the cardinality of $\bar{x}$ is k:

$$\bar{x} \ \in \mathbb{R}^N \quad \text{with } \|\bar{x}\|_0 = k \ (\ll N) \tag{3.2}$$

However, most signals themselves are not sparse but can be expressed sparsely in terms of a certain basis. This kind of signal is called a compressible signal, which is mathematically expressed as follows:

$$\bar{x} = \mathbf{\Psi}\bar{\gamma} \ \in \mathbb{R}^N \quad \text{with } \|\bar{\gamma}\|_0 = k \ (\ll N) \tag{3.3}$$

The matrix $\mathbf{\Psi}$ is typically a square matrix and called a sparsifying matrix. With a closer look at Eq.(3.3), the signal $\bar{x}$ is expressed as a linear combination of a few columns of Ψ chosen by $\bar{\gamma}$. In a signal processing field, this sparsifying matrix is also called a dictionary, and its columns are called atoms. Therefore, the signal $\bar{x}$ is a linear combination of atoms of the dictionary $\mathbf{\Psi}$.

Equating Eq.(3.1) and Eq.(3.3), a measurement of a compressible signal can be expressed as follows:

$$\begin{aligned} \bar{y} &= \mathbf{\Phi}\mathbf{\Psi}\bar{\gamma} \\ &= \mathbf{D}\bar{\gamma} \quad \text{with } \|\bar{\gamma}\|_0 \end{aligned} \tag{3.4}$$

Discrete cosine transform

One of the most well-known dictionaries is a Discrete Cosine Transform (DCT) matrix. DCT matrix can represent most natural images sparsely and hence it is often used for image compression.

Atoms of the $N \times N$ DCT matrix are the basis elements of the 2D-DCT and they are defined as follows [37]:

$$\mathbf{\Psi}_2(k,n) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 1,\ 1 \leq n \leq N \\ \sqrt{\frac{2}{N}} \cos \frac{\pi(2n-1)(k-1)}{2N}, & 2 \leq k \leq N,\ 1 \leq n \leq N \end{cases} \tag{3.5}$$

Suppose $\bar{x} \in \mathbb{R}^N$ is a vectorized form of an image $\bar{X} \in \mathbb{R}^{n \times n}$, it is expressed as a product between 2D-DCT matrix and a sparse coefficient vector $\bar{\gamma}$:

$$\bar{x} = \mathbf{\Psi}_2 \bar{\gamma} \tag{3.6}$$

Since 2D-DCT matrix is an orthogonal matrix, the sparse representation can be obtained by multiplying the transpose of $\mathbf{\Psi}_2$:

$$\bar{\gamma} = \mathbf{\Psi}_2^T \bar{x} \tag{3.7}$$

However, this operation is computationally expensive for large images since the size of Ψ_2 is proportional to the square of the image size. The alternative way is first applying the 1D-DCT matrix $\mathbf{\Psi}_1$ to all the columns of an image, then to all the rows of the result. This operation is much simpler and faster than Eq.(3.6).

$$\mathbf{\Gamma} = \mathbf{\Psi}_1^T \mathbf{X} \mathbf{\Psi}_1 \tag{3.8}$$

where $\mathbf{\Gamma}$ is a matrix form of $\bar{\gamma}$. The 1D-DCT matrix $\mathbf{\Phi}_1$ is obtained by arranging all the basis elements of 1D-DCT in a column-wise way.

The same computational shortcut is also possible for three dimensional signals. In this case, 1D-DCT is first applied to the columns, then to the rows, and finally to the width of the signal.

Sensing matrix

A sensing matrix is a key concept in compressed sensing for a stable reconstruction. The measurement by a sensing matrix $\mathbf{\Phi}$ in Eq.(3.4) must not damage the signal. However, in general, the sensing matrix Φ do damage the signal because it has larger number of columns than that of rows, which makes Eq.(3.4) an underdetermined problem with infinitely many solutions.

However, here the sparsity of signals comes into play. In order to evaluate stability of a solution of Eq.(3.4), a measure called Restricted Isometry Property (RIP) [38] is often used:

Definition 3.1.1 *For a matrix* $\mathbf{D}$ *of size* $m \times N$ $(N > m)$ *with* L_2*-normalized columns, and for an integer scalar* $s \leq m$*, consider sub-matrices* $\mathbf{D}_s$ *containing* s *columns from* $\mathbf{D}$*. Define* δ_s *as the smallest quantity such that*

$$\forall c \in \mathbb{R}^s \quad 1 - \delta_s \leq \frac{\|\mathbf{D}_s \bar{c}\|_2^2}{\|\bar{c}\|_2^2} \leq 1 + \delta_s \tag{3.9}$$

hold true for any choice of s *columns. Then* D *is said to have an* s*-RIP with a constant* δ_s*.*

If the matrix $\mathbf{D}$ in Eq.(3.4) has a k-RIP with a constant δ_k, nearly exact reconstruction is guaranteed. The key idea is that any subset of k columns from $\mathbf{D}$ transforms an arbitrary vector of size $k \times 1$ without almost any loses or gains of information.

Stability of the solution can be evaluated as follows. As an example, two sparse vectors, $\bar{\gamma}_0$ and $\hat{\gamma}$ are considered. A vector $\bar{\gamma}_0$ has a cardinality of k_0 and its multiplication with $\mathbf{D}$ is equal to $\bar{y}$ within the error of ε:

$$\|\bar{y} - \mathbf{D}\bar{\gamma}_0\|_2 \leq \varepsilon \tag{3.10}$$

On the other hand, a vector $\hat{\gamma}$ is a candidate solution of Eq.(3.4). Since the solution for Eq.(3.4) is obtained as a sparsest solution as it is discussed in the next section, the vector $\hat{\gamma}$ has k_0 nonzeros at most. Assume that this solution also satisfies Eq.(3.4) within the error of ε:

$$\|\bar{y} - \mathbf{D}\hat{\gamma}\|_2 \leq \varepsilon \tag{3.11}$$

Now the difference between the two sparse vectors is defined: $\Delta\bar{\gamma} = \hat{\gamma} - \gamma_0$. The goal is to show that L_2-norm of $\Delta\bar{\gamma}$ is small enough to be negligible. It can be readily shown from Eq.(3.10) and Eq.(3.11) that the product of $\mathbf{D}$ and $\Delta\bar{\gamma}$ is upperbounded by 2ε:

$$\|\mathbf{D}\bar{\gamma}_0 - \mathbf{D}\hat{\gamma}\|_2 = \|\mathbf{D}\Delta\bar{\gamma}\|_2 \leq 2\varepsilon \tag{3.12}$$

Moreover, $\Delta\bar{\gamma}$ is a sparse vector with $2k_0$ nonzeros at most:

$$\|\bar{\gamma}_0\|_0 + \|\hat{\gamma}\|_0 \leq 2k_0 \tag{3.13}$$

Since a matrix $\mathbf{D}$ satisfies the RIP property, it follows from Eq.(3.9) and Eq.(3.12) that

$$(1 - \delta_{2k_0})\|\Delta\bar{\gamma}\|_2^2 \leq \|\mathbf{D}\Delta\bar{\gamma}\|_2^2 \leq 4\varepsilon^2 \tag{3.14}$$

Therefore, stability of the solution can be evaluated as follows:

$$\|\Delta\bar{\gamma}\|_2^2 = \|\hat{\gamma} - \bar{\gamma}_0\|_2^2 \leq \frac{4\varepsilon^2}{1 - 2\delta_{2_0}} \tag{3.15}$$

Thus, if the matrix $\mathbf{D}$ in Eq.(3.4) satisfies the RIP, the solution is shown to be stable.

The next important point is how to design a matrix $\mathbf{D}$ so that it satisfies the RIP. Unfortunately, it is almost impossible to merely verify that a given matrix satisfies the RIP because of too many possible choices for the sub-matrices $\mathbf{D}_s$ in Eq.(3.9).

Surprisingly, Candes [39] showed that a randomness in the matrix guarantees the RIP. One of the most common matrices in compressed sensing is a Gaussian matrix whose entries are independent and identically distributed random variables from a zero-mean, $1/N$-variance Gaussian density. It is useful because if a matrix Φ is a Gaussian matrix, then the matrix $\mathbf{D} = \mathbf{\Psi\Phi}$ is also a Gaussian matrix regardless of the choice of $\mathbf{\Psi}$. In this respect, hereinafter this research will use a Gaussian matrix as a sensing matrix $\mathbf{\Phi}$.

Reconstruction

In compressed sensing, a compressible signal $\bar{x}$ can be reconstructed from its linear measurement $\bar{y}$ by seeking the sparsest solution of Eq.(3.4).

$$\hat{\gamma} = \text{argmin}\|\bar{\gamma}\|_0 \quad \text{s.t. } \bar{y} = \mathbf{D}\bar{\gamma} \tag{3.16}$$

$$\hat{x} = \Psi\hat{\gamma} \tag{3.17}$$

This problem is often referred to as a P0 problem. It can be solved by several methods which will be discussed later.

This P0 problem is the most straightforward way to express compressed sensing in a nutshell. However, it is not the ideal problem to target in practical cases. First of all, the equality constraint is too strict in practical cases. If the measurement $\bar{y}$ is slightly perturbed by noise, the system $\bar{y} = \mathbf{D}\bar{\gamma}$ will not have a sparse solution at all. Moreover, the L_0 measure is also too strict. Even a slight random perturbation of $\bar{\gamma}$ makes the solution not sparse at all.

There is a more robust alternative for this P0 problem:

$$\hat{\gamma} = \text{argmin}\|\bar{\gamma}\|_0 \quad \text{s.t. } \|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 \leq \varepsilon^2 \tag{3.18}$$

$$\hat{x} = \mathbf{\Psi}\hat{\gamma} \tag{3.19}$$

This is referred to as a P0-epsilon problem. Although this problem is an approximation of the P0 problem, it has more benefits. This problem allows ε-deviation between $\bar{y}$ and $\mathbf{D}\bar{\gamma}$, which makes it robust to the perturbation of $\bar{y}$. The same term also absorbs the perturbation of the solution if the norm of the noise vector is less than 1. This is explained as follows.

Suppose the true solution $\bar{\gamma}_0$ is perturbed by noise:

$$\bar{\gamma} = \bar{\gamma}_0 + \varepsilon\ \bar{u} \quad \text{with } \varepsilon \ll 1 \text{ and } \|\bar{u}\|_2 = 1 \tag{3.20}$$

Then, the L_2 norm of the residual can be evaluated as follows.

$$\begin{aligned} \|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 &= \|\bar{y} - \mathbf{D}(\bar{\gamma}_0 + \varepsilon\bar{u})\|_2^2 \\ &= \|(\bar{y} - \mathbf{D}\bar{\gamma}_0) - \varepsilon\mathbf{D}\bar{u})\|_2^2 \\ &= \varepsilon\|\mathbf{D}\bar{u}\|_2^2 \ \leq \ \varepsilon^2 \end{aligned} \tag{3.21}$$

Therefore, this P0-epsilon problem can return the solution even with a slight perturbation. Thus, the ε-deviation solves the both issues of P0 problem.

Reconstruction algorithm: OMP algorithm

There are mainly two methods to solve the P0 problem and the P0-epsilon problem: greedy methods and relaxation methods. The Greedy methods build a solution by choosing one nonzero element at a time, while the relaxation methods approximate the problem as a continuous optimization problem.

Here, the OMP algorithm [23] is introduced as an example of Greedy methods [38]. The biggest advantage of the OMP is that it can solve P0-epsilon problem as well as P0 problem. There are two types of equivalent forms of the P0-epsilon problem. One is a sparsity-constraint problem (Eq.(3.22)) and the other is an error-constraint problem (Eq.(3.23)).

$$\hat{\gamma} = \text{argmin } \|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 \quad \text{s.t. } \|\bar{\gamma}\|_0 = K \tag{3.22}$$

$$\hat{\gamma} = \text{argmin } \|\bar{\gamma}\|_0 \qquad \text{s.t. } \|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 \leq \varepsilon^2 \tag{3.23}$$

The core idea of the OMP algorithm is to approximate the vector $\bar{y}$ as a linear combination of the column vectors of $\mathbf{D}$ (see Algorithm 1).

Algorithm 1 Orthogonal Matching Pursuit [23]

Input : Dictionary $\mathbf{D}$, signal $\bar{y}$, target sparsity K or target error ε

Output : Sparse representation $\bar{\gamma}$ such that $y \approx \mathbf{D}\bar{\gamma}$

Initialization: Set $I := (), \bar{r} := \bar{y}, \bar{\gamma} := \bar{0}$

1 **while** *(stopping criterion not met)* **do**

2 $\quad \hat{i} := \text{argmax } \|\bar{d}_i^T \bar{r}\|$

3 $\quad I := (I, \hat{i})$

4 $\quad \bar{\gamma}_I := (\mathbf{D}_I)^+ \bar{y}$

5 $\quad \bar{r} := \bar{y} - \mathbf{D}_I \bar{\gamma}_I$

6 **end while**

First, the OMP finds an atom with the largest inner product between the current residual (line 2) and update the indices I (line 3). Subsequently, the signal is orthogonally projected to the chosen atoms (line 4), and the residual is computed again (line 5). This process is repeated until either the sparsity goal or the error goal is met. Note that given a sequence of indices $I = (i_1, i_2, ..., i_k)$ and a matrix $\mathbf{M}$, a matrix $\mathbf{M}_I$ is a sub-matrix of $\mathbf{M}$ consisting of columns indexed by I. Similarly, for a vector, $\bar{v}_I$ is a sub-vector of a vector $\bar{v}$.

Reconstruction algorithm: Batch-OMP algorithm

This paper uses a batch-OMP algorithm [23] because of its fast computation time. This algorithm is a modified version of the OMP algorithm.

The batch-OMP lowers the computational cost of the OMP by utilizing two key methods. The first method is an implementation of a progressive Cholesky update process to avoid an explicit computation of an inversion of $\mathbf{D}_I^T \mathbf{D}_I$ in line 4 of Algorithm 1. The second method is a pre-computation of $\bar{\alpha}^0$ and $\mathbf{G}$. This circumvents an explicit computation of the residual $\bar{r}$ in line 5 of Algorithm 1. These modifications give a substantial improvement in computational complexity. Suppose that a dictionary size

is $N \times L$ and a target sparsity is K, then the complexities of the two algorithms are as follows:

$$T_{omp} = K^3 + 2KNL \tag{3.24}$$

$$T_{b-omp} = K^3 + K^2L + 2NL \tag{3.25}$$

For example, if $K = \sqrt{(N)}/2$ and $L = 2N$, then they are computed as:

$$T_{omp} = 2N^{2.5} \tag{3.26}$$

$$T_{b-omp} = 4.5N^2 \tag{3.27}$$

This computation does not take into account the time required to pre-compute $\mathbf{G}$ but it becomes insignificant as the number of signals increases. Rubinstein points out that for $N = 256$, the ratio between the two methods approaches to 7.11 for example:

$$\frac{T_{omp}}{T_{b-omp}} = \frac{2}{4.5}\sqrt{N} \approx 7.11 \tag{3.28}$$

Algorithm 2 Batch-OMP [23]

Input : $\bar{\alpha}^0 = D^T\bar{y}, \varepsilon^0 = \bar{y}^T\bar{y}, \mathbf{G} = \mathbf{D}^T\mathbf{D}$, target error ε

Output : Sparse representation $\bar{\gamma}$ such that $\bar{y} \approx \mathbf{D}\bar{\gamma}$

Initialization: Set $I := (),\ \mathbf{L}^1; = [1],\ \delta^0 := 0,\ n := 1$

while $\varepsilon^{n-1} > \varepsilon$ **do**

  $\hat{k} := argmax\ \|\alpha_k^{n-1}\|$

  $\bar{g} := \mathbf{G}_{I^n,\hat{k}}$

  if $n > 1$ **then**

    $\bar{w} := Solve\ for\ \bar{w}\ \{\mathbf{L}^{n-1}\bar{w} = \bar{g}_{I_n}\}$

    $\mathbf{L}^n := \begin{bmatrix} \mathbf{L}^{n-1} & 0 \\ \bar{w}^T & \sqrt{1 - \bar{w}^T\bar{w}} \end{bmatrix}$

  end if

  $I^n := (I^{n-1}, \hat{k})$

  $\bar{c}^n := Solve\ for\ \left\{\mathbf{L}^n(\mathbf{L}^n)^T\bar{c} = \bar{\alpha}^0_{I^n}\right\}$

  $\bar{\beta}^n = \mathbf{G}_{I^n}\bar{c}^n$

  $\bar{\alpha}^n := \bar{\alpha}^0 - \bar{\beta}^n$

  $\delta^n = (\bar{c}^n)^T\bar{\beta}^n_{I^n}$

  $\varepsilon^n = \varepsilon^{n-1} - \delta^n + \delta^{n-1}$

  $n := n + 1$

end while

$\bar{\gamma} := 0$

$\bar{\gamma}_{I^n} := \bar{c}^n$

Reconstruction algorithm: ADMM algorithm

While the OMP and batch-OMP algorithms are categorized into greedy methods, the ADMM (Alternating Direction Method of Multipliers) algorithm [40] belongs to the group of relaxation methods.

In relaxation methods, the P0-epsilon problem is approximated by the L_1-minimization problem:

$$\hat{\gamma} = \text{argmin } \|\bar{\gamma}\|_1 \quad \text{s.t. } \|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 \leq \varepsilon^2 \tag{3.29}$$

The constraint in Eq.(3.29) is often turned into a penalty by using a Lagrange multiplier λ:

$$\hat{\gamma} = \text{argmin } \lambda\|\bar{\gamma}\|_1 + \frac{1}{2}\|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 \tag{3.30}$$

On the other hand, the ADMM algorithm deals with the following type of optimization problem:

$$\hat{x} = \text{argmin } f(\bar{x}) + g(\bar{x}) \tag{3.31}$$

where the two functions $f(\bar{x})$ and $g(\bar{x})$ are easy to minimize individually but the function $f(\bar{x}) + g(\bar{x})$ is difficult to deal with. This problem can be restated by introducing a new variable $\bar{v}$:

$$\hat{x} = \text{argmin } f(\bar{x}) + g(\bar{v}) \quad \text{s.t. } \bar{x} = \bar{v} \tag{3.32}$$

Introducing a Lagrangian multiplier vector $\bar{u}$ and turning the constraint into penalty, the Augumented-Lagrangian can be formed as follows:

$$L(\bar{x}, \bar{v}, \bar{u}) = f(\bar{x}) + g(\bar{x}) + \frac{1}{2}\|\bar{x} - \bar{v}\|_2^2 + \bar{u}^T(\bar{x} - \bar{v}) \tag{3.33}$$

$$= f(\bar{x}) + g(\bar{x}) + \frac{1}{2}\|\bar{x} - \bar{v} + \bar{u}\|_2^2 \tag{3.34}$$

Using this Augumented-Lagrangian, Eq.(3.32) can be expressed as a set of optimization problems as follows:

$$\bar{x}_k = \text{argmin } f(\bar{x}) + \frac{1}{2}\|\bar{x} - \bar{v}_{k-1} + \bar{u}_{k-1}\|_2^2 \tag{3.35}$$

$$\bar{v}_k = \text{argmin } g(\bar{v}) + \frac{1}{2}\|\bar{x}_k - \bar{v} + \bar{u}_{k-1}\|_2^2 \tag{3.36}$$

$$\bar{u}_k = \bar{u}_{k-1} + \bar{x}_k - \bar{v}_k \tag{3.37}$$

Therefore, using Eq.(3.35), (3.36) and (3.37), the problem (3.30) can be written as follows:

$$\bar{\gamma}_k = \text{argmin } \|\bar{\gamma}\|_1 + \frac{1}{2}\|\bar{\gamma} - \bar{v}_{k-1} + \bar{u}_{k-1}\|_2^2 \tag{3.38}$$

$$\bar{v}_k = \text{argmin } \frac{1}{2}\|\bar{y} - \mathbf{D}\bar{\gamma}\|_2^2 + \frac{1}{2}\|\bar{\gamma}_k - \bar{v} + \bar{u}_{k-1}\|_2^2 \tag{3.39}$$

$$\bar{u}_k = \bar{u}_{k-1} + \bar{\gamma}_k - \bar{v}_k \tag{3.40}$$

In this way, the problem (3.30) becomes much easier to solve.

First, Eq.(3.38) has a simple quadratic form which has a closed form solution:

$$\bar{\gamma}_k = (\mathbf{D}^T\mathbf{D} + \bar{I})^{-1}(\mathbf{D}^T\bar{b} + \bar{v}_{k-1} - \bar{u}_{k-1}) \tag{3.41}$$

Since a matrix $\mathbf{D}^T\mathbf{D} + \bar{I}$ is a symmetric positive-definite matrix, Eq.(3.38) can be easily computed by Cholesky-factorization. On the other hand, it is known that the problem (3.39) is solved by an algorithm called soft-thresholding [40].

$$\bar{v}_k = S_\lambda(\bar{\gamma}_k + \bar{u}_{k-1}) \tag{3.42}$$

$$= \begin{cases} \bar{\gamma}_k + \bar{u}_{k-1} + \lambda & \bar{\gamma}_k + \bar{u}_{k-1} \leq -\lambda \\ 0 & \|\bar{\gamma}_k + \bar{u}_{k-1}\| < \lambda \\ \bar{\gamma}_k + \bar{u}_{k-1} + \lambda & \bar{\gamma}_k + \bar{u}_{k-1} \geq \lambda \end{cases} \tag{3.43}$$

Therefore, the overall structure of the ADMM algorithm is described in algorithm 3.

Algorithm 3 ADMM [40]

Input : Signal set $\bar{y}$, Dictionary $\mathbf{D}$, number of iterations n

Output : Sparse representation $\bar{\gamma}$ such that $\bar{y} \approx \mathbf{D}\bar{\gamma}$

Initialization: Set $\bar{\gamma}_0 := 0$, $\bar{v}_0 := 0$, $\bar{u}_0 := 0$, $k := 0$

1 **for** $k = 1, \ldots, n$ **do**

2 $\quad \bar{\gamma}_k = (\mathbf{D}^T\mathbf{D} + \bar{I})^{-1}(\mathbf{D}^T\bar{b} + \bar{v}_{k-1} - \bar{u}_{k-1})$

3 $\quad \bar{v}_k = S_\lambda(\bar{\gamma}_k + \bar{u}_{k-1})$

4 $\quad \bar{u}_k = \bar{u}_{k-1} + \bar{\gamma}_k - \bar{v}_k$

5 **end for**

3.1.2 Simulation

Now some examples of the compressed sensing are shown. In this simulation, an image is compressed and then reconstructed by the batch-OMP algorithm. Figure 3.1 shows the image used for this simulation. This image is called Barbara, which is often used as a benchmark in image processing field [38]. The size of the image is 128×128. The sensing matrix is a Gaussian matrix with mean 0 and variance $1/128^2$, and the compression ratio is 64%. The quality of the reconstructed image is evaluate based on PSNR (Peak Signal-to-Noise-Ratio) which is defined as follows:

$$\mathrm{PSNR} = 10 \cdot \log_{10} \left\{ \frac{\mathrm{imsize}^2 \cdot d^2}{\|\hat{x} - \bar{x}_0\|_2^2} \right\} \tag{3.44}$$

where imsize is an edge length of a square image and d is a dynamic range, which is a difference between the maximum value and the minimum value in the original image. $\bar{x}_0$ is the original image vector and $\hat{x}$ is the reconstructed image vector. The PSNR is a measure for the clearness of images.

Fig. 3.1.: Image Barbara

Fig. 3.2.: Compressed image and two reconstructed results (Result 1: sparsity-constraint with $K = 4000$, Result 2: error-constraint with $\varepsilon = 1$)

The reconstruction result is shown in Figure 3.2. The leftmost image shows the compressed image y in a matrix form. Given this compressed image and a sensing matrix $\mathbf{\Phi}$, two images are reconstructed. Result 1 shows a solution of a sparsity-constraint problem (Eq.(3.22)) with $K = 4000$ and Result 2 shows a solution of an error-constraint problem (Eq.(3.23)) with $\varepsilon = 1$. The calculation time is 358.8 seconds for Result 1 and 1718.6 seconds for Result 2. The simulations have been performed on a Dell Precision Tower 5810 computer equipped with an Intel Xeon E5-1650 and 16 GB of RAM running a 64-bit version of Windows 7.

The PSNR value of the result depends on the threshold K or ε. The K value is chosen so that the PSNR value becomes the maximum. Figure 3.3 shows the relationship between the value K and the PSNR of the result. This figure shows that the PSNR has become the maximum value when $K \approx 4,000$ and it gets saturated. On the other hand, the smaller ε results in better PSNR value and longer computation time. The value of ε is chosen so that the computation time is within 30 minutes.

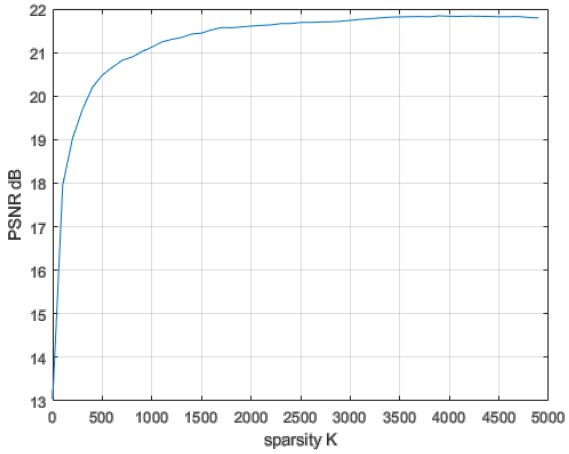


Fig. 3.3.: Relationship between the sparsity threshold and the PSNR of the result

The PSNR value of the result also depends on the compression ratio. Of course, a higher compression ratio improves the PSNR value of the result but also increases the computation time. Figure 3.4 shows the relationship between the compression ratio and the PSNR value. Empirically, the PSNR value needs to be greater than $\sim 20dB$ for an image to be recognizable. Therefore, the compression ratio needs to be at least around 50%.

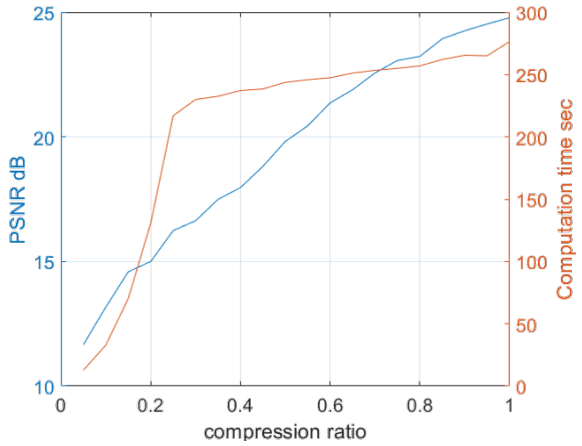


Fig. 3.4.: Relationship between the compression ratio and the PSNR/computation time (under sparsity constraint with $K = 4,000$)

3.2 Single-Pixel Camera

3.2.1 Overview

A single-pixel camera is one of the applications of compressed sensing suggested by Takhar et.al [41]. Figure 3.6 [41] shows a block diagram of a single-pixel camera. It has been used in a wide variety of fields including three-dimensional imaging [42] (Figure 3.5) and remote sensing technique [15].

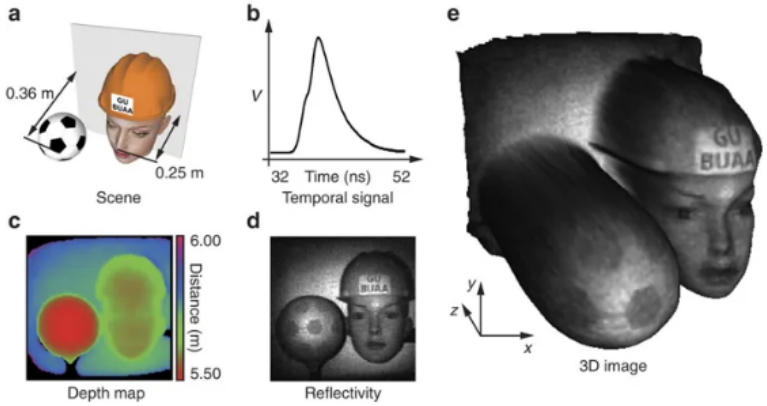


Fig. 3.5.: Three-dimensional imaging by a single-pixel camera [42]

Mathematical background

Prior to delving into the specifics of a single-pixel camera, it is prudent to recapitulate the measurement process in the compressed sensing framework in detail. As discussed in the previous section, an image vector $\bar{x} \in \mathbb{R}^{n^2}$ is measured as a product of a sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times n^2}$:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{3.45}$$

where n is a size of one edge of a square image. The k^{th} element of $\bar{y}$ is expressed as an inner product between a vector x and a k^{th} row of $\mathbf{\Phi}$:

$$y_k = \bar{\Phi}_k^T \, \bar{x} = \begin{bmatrix} \phi_{k1} & \dots & \phi_{kn^2} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n^2} \end{bmatrix} = \sum_{l=1}^{n^2} \phi_{kl} x_l \tag{3.46}$$

These vectors $\bar{\Phi}_k$ and $\bar{x}$ both have a length of n^2, and they are reshaped into matrices, $\tilde{\mathbf{\Phi}}_k$ and $\mathbf{X}$ respectively:

$$\tilde{\mathbf{\Phi}}_k = \begin{bmatrix} \phi_{k1} & \dots & \phi_{k(n^2-n+1)} \\ \vdots & \ddots & \vdots \\ \phi_{kn} & \dots & \phi_{kn^2} \end{bmatrix}, \; \mathbf{X} = \begin{bmatrix} x_1 & \dots & x_{n^2-n+1} \\ \vdots & \ddots & \vdots \\ x_n & \dots & x_{n^2} \end{bmatrix} \tag{3.47}$$

Since the size of the two matrices are the same, their element-wise product can be computed. The summation of the element-wise product between these two matrices are the same with the value of y_k:

$$y_k = \sum_{\text{all}} \tilde{\mathbf{\Phi}}_k \odot \mathbf{X} \tag{3.48}$$

Therefore, the measurement vector $\bar{y}$ in Eq.(3.45) can be obtained by getting the element-wise product m times. This idea is practiced in a single-pixel camera.

Mechanism of a single-pixel camera

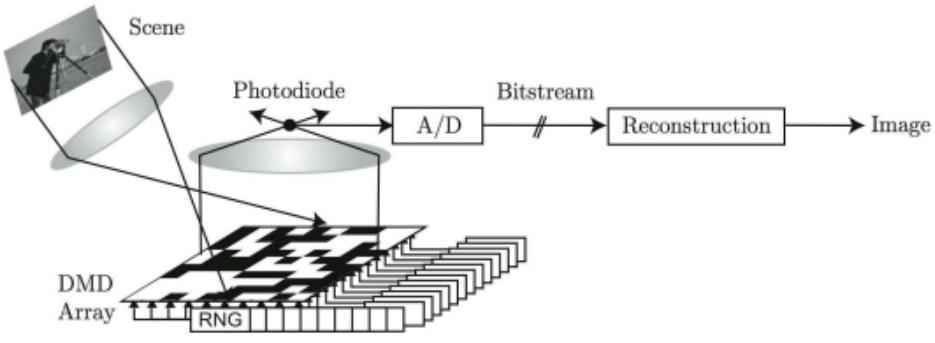


Fig. 3.6.: Single-pixel camera diagram [41], p.5

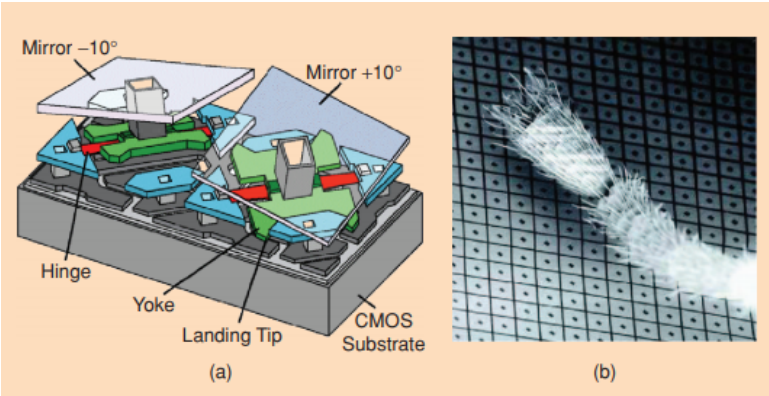


Fig. 3.7.: (a) Schematic of two mirrors of DMD. (b) A portion of an actual DMD array with an ant leg for scale [43], p.89

The mechanism of the camera can be explained as follows. First, an incident light field from the image of interest is collected by the first convex lens. This light field corresponds to a matrix $\mathbf{X}$ in Eq.(3.47). Subsequently, the light field is focused onto a digital micro-mirror device (DMD) consisting of millions of micro mirrors (Figure 3.7). The light flux from each pixel of the image reflects on a particular mirror of the DMD in a pixel-wise way. Each mirror can be oriented in two directions: -10^o or 10^o. If a light flux is reflected on a mirror orienting in 10^o, the light flux can be collected by the second lens but otherwise, the light cannot be collected. Therefore, these two directions of the mirror correspond to 0-output and 1-output respectively. All the light fluxes from the lens 1 are reflected on DMD with an output of either 0 or 1, and they are collected by lens 2. The total intensity of the light is measured by a single photo-diode, processed by A/D converter. This measurement process is repeated m times and the vector $\bar{y}$ is obtained. Finally, given $\bar{y}$, the image $\bar{x}$ is reconstructed by compressed sensing scheme.

Comparison with compressed sensing

These procedures correspond to compressed sensing. The DMD array corresponds to the matrix $\tilde{\mathbf{\Phi}}_k$ in Eq.(3.45) and the measurement of the total intensity of light in the single photo-diode corresponds to the calculation in Eq.(3.48). Each mirror in the DMD returns a product between a pixel value and either 0 or 1. The summation of all the products equals to the k^{th} element of $\bar{y}$. By repeating this measurement m times, the all elements of $\bar{y}$ can be obtained.

The only difference from the compressed sensing is that the elements of the sensing matrix is limited to either 0 or 1. However, by getting a random mirror pattern, this sensing matrix also satisfies the RIP condition and hence the reconstruction of an image is guaranteed. The most simple pattern that can be generated by DMD is a Rademacher pattern:

$$\phi_{ij} = \begin{cases} 1 & p = 1/2 \\ -1 & p = 1/2 \end{cases} \tag{3.49}$$

where p is a probability. Baraniuk [44] showed that a sensing matrix generated by this pattern satisfies the RIP condition with high probability. Although the DMD returns only 0 or 1, the Rademacher pattern can be realized by a simple calculation:

$$\begin{array}{ccccccccc} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ & & & & \downarrow \times 2 & & & & \\ 0 & 2 & 0 & 2 & 2 & 2 & 0 & 0 & 2 \\ & & & & \downarrow -1 & & & & \\ \text{-1} & 1 & \text{-1} & 1 & 1 & 1 & \text{-1} & \text{-1} & 1 \end{array}$$

First, an image is measured by using a sensing matrix whose element is 1 by probability of 1/2 and 0 by probability of 1/2. Subsequently, the result of the measurement is multiplied by 2. An image is again measured by a sensing matrix whose elements are all 1. By substituting the second result from the first result, an image can be measured by the Rademacher pattern [45].

It is also possible to get a Gaussian matrix on the mirror patterns by using a similar technique called Pulse-width modulation (PWM). Before implementing the PWM, all the elements in the Gaussian matrix needs to be modified to minimize the numerical error as much as possible. First, all the values are offset so that the minimum value becomes zero. Subsequently, they are stretched so that the maximum value becomes one. Now everything is ready for the PWM. This technique enables to approximately get an arbitrary value between 0 and 1. In each time sequence, the mirror can only return 0 or 1. However, by adjusting the number of outputs during a certain time frame, it gets a decimal value as an average. For example, in Figure 3.8, during a time frame of $5t$, a mirror returns an output of 1 for only t period of time. In this case, the average signal value during this time frame is 0.2. Therefore, this method enables us to measure an image by a sensing matrix whose elements are between 0 and 1. After all the measurements are obtained, they are modified so that they become equivalent with being measured by a Gaussian matrix. Since the measurement is a linear operation, the modification is done easily.

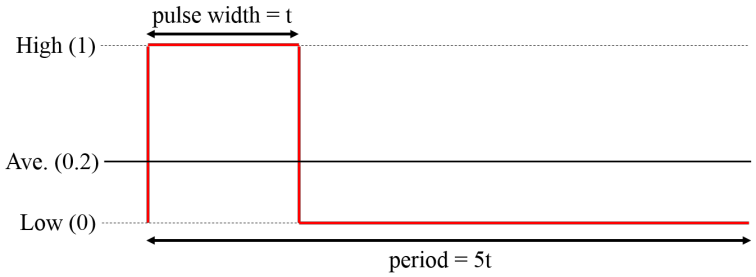


Fig. 3.8.: How to express a positive value less than 1 by PWM

One problem of this approach is a trade-off between an accuracy and an exposure time. The accuracy of the approximation increases with longer digits of the decimal number, which needs a longer time frame. Suppose a signal value needed is 0.1234 and let n_p denote the number of measurements to approximate this value. If $n_p = 10^2$, 12 times out of 10^2 times are the output of 1 and a signal value obtained is 0.12. On

the other hand, if $n_p = 10^4$, a signal value obtained is 0.1234, which is by far more accurate but takes longer time. Since measurement time increases by an order of 10^n to get n-times more accurate results, it is necessary to consider how much round-off error is allowed in this measurement process.

3.2.2 Simulation

Algorithm

As mentioned earlier, the only difference between the compressed sensing and single-pixel camera in a simulation is a way to obtain the measurement. Here, the mathematical expression for a single-pixel camera and its relationship with compressed sensing are discussed. The goal is to approximate the measurement vector $\bar{y}$ which can be obtained by:

$$\text{Goal:} \quad \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{\bar{y}} = \underbrace{\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \dots & \phi_{1N} \\ \phi_{21} & \dots & & & \\ \vdots & & & & \\ \phi_{m1} & \phi_{m2} & \dots & \dots & \phi_{mN} \end{bmatrix}}_{\mathbf{\Phi}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}}_{\bar{x}} \tag{3.50}$$

where $\mathbf{\Phi}$ is a Gaussian matrix and $\bar{x}$ is a vector form of the observed image. First, all the elements of matrix $\mathbf{\Phi}$ are offset by its smallest element ϕ_{min}. Subsequently, all the elements are stretched by coefficient $k = 1/\phi_{max}$, where ϕ_{max} is the largest element after the offset. As a result, a matrix $\mathbf{\Phi}_{\text{DMD}}$ whose elements are between 0 and 1 is obtained:

$$\mathbf{\Phi}_{\text{DMD}} = \begin{bmatrix} k(\phi_{11} - \phi_{min}) & \dots & \dots & k(\phi_{1N} - \phi_{min}) \\ k(\phi_{21} - \phi_{min}) & \dots & & \\ \vdots & & & \\ k(\phi_{m1} - \phi_{min}) & \dots & \dots & k(\phi_{mN} - \phi_{min}) \end{bmatrix} \equiv \begin{bmatrix} \tilde{\phi}_{11} & \dots & \dots & \tilde{\phi}_{1N} \\ \tilde{\phi}_{21} & \dots & & \\ \vdots & & & \\ \tilde{\phi}_{m1} & \dots & \dots & \tilde{\phi}_{mN} \end{bmatrix} \tag{3.51}$$

The next step is to approximate the elements of $\mathbf{\Phi}_{\mathrm{DMD}}$ by PWM. Each element $\tilde{\phi}_{ij}$ $(1 \leq i \leq m,\ 1 \leq j \leq N)$ is expressed as a time average of mirror sequences. Let ρ_{ij}^{t} denote a mirror value corresponding to an element $\tilde{\phi}_{ij}$ at time t. Its value is either zero or one.

$$\rho_{ij}^{t} = 0 \text{ or } 1, \tag{3.52}$$

$$\tilde{\phi}_{ij} \simeq \frac{1}{p} \sum_{t=t_1}^{t_p} \rho_{ij}^{t} \tag{3.53}$$

For example, if $\tilde{\phi}_{ij} = 0.2$ is a target value of the element and $p = 5$, then the target mirror pattern is $\left\{\rho_{ij}^{t_1},\ \rho_{ij}^{t_2},\ \rho_{ij}^{t_3},\ \rho_{ij}^{t_4},\ \rho_{ij}^{t_5}\right\} = \{0,\ 0,\ 0,\ 0,\ 1\}$.

In a single-pixel camera, an image or scene is measured m times and each measurement corresponds to the i^{th} row of $\mathbf{\Phi}_{\mathrm{DMD}}$ in Eq.(3.51). From Eq.(3.51) and Eq.(3.53), the i^{th} measurement obtained by DMD is expressed as follows:

$$y_{\mathrm{DMD}}(i) = \frac{1}{p} \sum_{j=1}^{N} \sum_{t=t_1}^{t_p} \rho_{ij}^{t}\ x_i \tag{3.54}$$

$$\simeq \sum_{j=1}^{N} \tilde{\phi}_{i,j}\ x_i \tag{3.55}$$

However, this value cannot be directly used for reconstruction algorithm and needs modification because of the offset and stretch operation which is done earlier. From Eq.(3.51), the relationship between the true Gaussian element and the stretched element is given as follows:

$$\tilde{\phi_{ij}} = k(\phi_{ij} - \phi_{min}) \tag{3.56}$$

Substitution of Eq.(3.56) into Eq.(3.55) gives

$$y_{\mathrm{DMD}}(i) = \sum_{j=1}^{N} k(\phi_{ij} - \phi_{min})\ x_i \tag{3.57}$$

$$= k \sum_{j=1}^{N} \phi_{ij} x_i - \phi_{min} \sum_{j=1}^{N} x_i \tag{3.58}$$

From Eq.(3.50), the i^{th} element of the true measurement is expressed as

$$y(i) = \sum_{j=1}^{N} \phi_{ij} x_i \tag{3.59}$$

Therefore, equating Eq.(3.58) and Eq.(3.59), the true measurement is expressed in terms of the DMD measurement as follows:

$$y(i) = \frac{1}{k}\, y_{\mathrm{DMD}}(i) + \phi_{min} \sum_{j=1}^{N} x_i \tag{3.60}$$

Therefore, in a simulation of a single-pixel camera, a measurement of an image by DMD is simulated by using Eq.(3.54). Subsequently, the obtained value is modified to a correct value by Eq.(3.60).

Simulation setup and Results

Based on this mathematics, single-pixel camera observation is simulated by using two different mirror patterns: the Rademacher pattern and the Gaussian pattern. In both mirror patterns, the Barbara image (Figure 3.1) of size 100×100 is sensed by a sensing matrix of size 6400×10000. In compressed sensing, this is directly computed as a product between the image vector and the sensing matrix. However, in a single-pixel camera, the image is measured by each row of the sensing matrix independently. Each row is approximated by n_d mirror sequences and this is repeated for 6400 times. Therefore, a required exposure time in the measurement process can be estimated by the following formula:

$$t_{ex} = t_u \times n_p \times 6400 \tag{3.61}$$

Note that t_u is an exposure time for one mirror pattern, which is assumed to be $1\mu s$. The value of n_p depends on a kind of mirror pattern and a level of approximation. If the Rademacher pattern is chosen, it always follows that

$$n_p = 1 \tag{3.62}$$

since only one mirror pattern corresponds to each row of the sensing matrix. On the other hand, if the Gaussian pattern is chosen, the value of n_p depends on how well the element of the sensing matrix is approximated:

$$n_p = 10^d \tag{3.63}$$

where d is a number of significant digits guaranteed by PWM. The image is reconstructed by batch-OMP algorithm under sparsity constraint where $K = 4,000$.

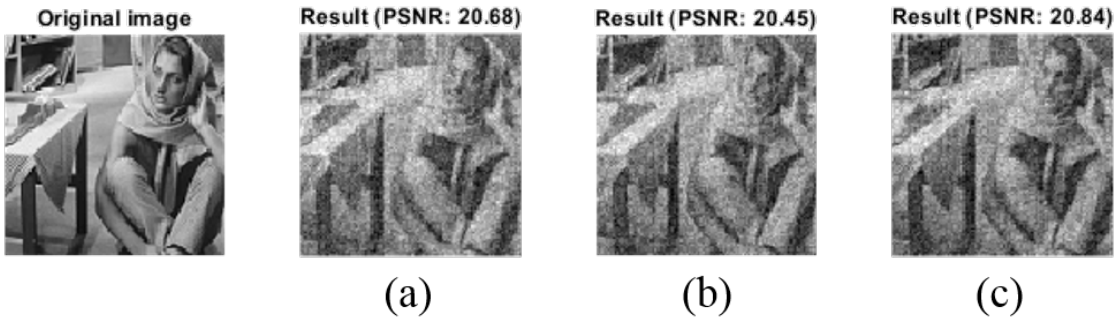


Fig. 3.9.: Images reconstructed by single-pixel camera based on different mirror patterns or exposure time (See Table for more details)

Table 3.1.: Mirror patterns and parameters in Figure 3.9

	Result (a)	Result (b)	Result (c)
Mirror pattern	Rademacher	Gaussian	Gaussian
Required exposure time	6.4×10^{-3} min	64 min	640 min

Figure 3.9 shows the results reconstructed under different mirror patterns and exposure times described in Table 3.1. Clearly, the Rademacher pattern has an advantage in the exposure time. However, the Gaussian pattern gives better PSNR values with longer exposure time because of the better approximation of the sensing matrix.

4. DICTIONARY LEARNING

4.1 Background

In a sparse-land model [38], it is assumed that most signals are represented by a simple linear system:

$$\bar{x} = \mathbf{D}\bar{\gamma} \quad \text{with } \|\bar{\gamma}\|_0 = k \ (\ll n_{\text{pixel}}) \tag{4.1}$$

where $\bar{x}$ is a signal vector of length n_{pixel}, $\mathbf{D}$ is a matrix and $\bar{\gamma}$ is a sparse vector. In this model, the signal $\bar{x}$ is represented by a linear combination of only k columns of $\mathbf{D}$. Therefore, "one can consider $\mathbf{D}$ as the periodic table of the fundamental elements in chemistry to describe the signal [38]". In this case, the matrix $\mathbf{D}$ is called a *dictionary* of the signal $\bar{x}$ and its columns are called *atoms*. For examples, a discrete cosine matrix Ψ is a most common dictionary for natural images as already discussed in section 3.1:

$$\bar{x} = \mathbf{\Psi}\bar{\gamma} \quad \text{with } \|\bar{\gamma}\|_0 = k \ (\ll N) \tag{4.2}$$

In this case, DCT helps to compress the image signal $\bar{x}$.

The main focus of attention in an image processing community is how to design a dictionary to better describe images accurately with the sparsest possible coefficients. One of the most straightforward approaches is to choose a pre-constructed dictionary that best fits a signal of interest. There exist some well-known dictionaries including Fourier, DCT, Hadamard [46], Wavelet [47], Curvelet [48] and Contourlet [49], most of which correspond to an inverse transform. Another approach is to adjust parameters of a certain adaptable dictionaries to fit into the signal. The most famous examples are Wavelet packets [50] and bandlets [51]. However, these dictionaries are restricted to a certain type of images. A dictionary which adapts to any kinds of images can be

found by introducing a new aspect, learning. This is a basic motivation of dictionary learning.

The dictionary learning starts with a training database of images of interest $\{\bar{x}_i\}_{i=1}^{N}$. This training database can be totally different types of images, similar type of images [52] or cropped patches of one image. Based on this database, a common dictionary $\mathbf{D}$ is found so that it describes all the images in the database as accurately as possible:

$$\bar{x}_i = \mathbf{D}\bar{\gamma}_i + \bar{v}_i \quad \text{with } \|\bar{\gamma}_i\|_0 = k_0,\ \|\bar{v}_i\|_2 \leq \varepsilon,\ i = 1, ..., N \tag{4.3}$$

where $\bar{v}_i$ is a representation error vector of the i^{th} image and ε is an upper bound of the representation error. This learning objective can be formulated as one of the following optimization tasks:

$$\{\hat{\mathbf{D}}, \hat{\gamma}_i\} = \text{argmin } \|\gamma_i\|_0 \quad \text{s.t. } \forall i,\ \|\bar{x}_i - \mathbf{D}\bar{\gamma}_i\|_2 \leq \varepsilon \tag{4.4}$$

$$\{\hat{\mathbf{D}}, \hat{\gamma}_i\} = \text{argmin } \|\bar{x}_i - \mathbf{D}\bar{\gamma}_i\|_2 \quad \text{s.t. } \forall i,\ \|\bar{\gamma}_i\|_0 \leq T \tag{4.5}$$

Note that the training database and its sparse representation are also represented in matrices form, $\mathbf{X}$ and $\mathbf{\Gamma}$ whose columns are $\bar{x}_i$ and $\bar{\gamma}_i$ respectively. Using this matrix representation, Eq.(4.4) and Eq.(4.5) are expressed as follows:

$$\{\hat{\mathbf{D}}, \hat{\mathbf{\Gamma}}\} = \text{argmin } \|\gamma_i\|_0 \quad \text{s.t. } \|\mathbf{X} - \mathbf{D}\mathbf{\Gamma}\|_F^2 \leq \varepsilon \tag{4.6}$$

$$\{\hat{\mathbf{D}}, \hat{\mathbf{\Gamma}}\} = \text{argmin } \|\mathbf{X} - \mathbf{D}\mathbf{\Gamma}\|_F^2 \quad \text{s.t. } \forall i,\ \|\bar{\gamma}_i\|_0 \leq T \tag{4.7}$$

This optimization problem is clearly not a well-posed problem since a permutation of atoms in $\hat{\mathbf{D}}$ does not affect the solution if the corresponding elements in $\hat{\gamma}_i$ are also permuted. Moreover, the scale between $\mathbf{D}$ and $\bar{\gamma}_i$ is not defined. In many cases, this issue is fixed by setting a constraint such that $\mathbf{D}$ has a normalized atoms:

$$\text{diag}\left\{\mathbf{D}^T\mathbf{D}\right\} = \mathbf{I} \tag{4.8}$$

where diag$\{\cdot\}$ represents diagonal elements of a matrix and $\mathbf{I}$ is an identity matrix.

4.2 Algorithms

There have been large number of algorithms for solving this dictionary learning problem. Here, the KSVD (K-singular value decomposition) algorithm [23] [53] and its modified version, sparse-KSVD algorithm [19] are introduced. Both of these algorithms are in the sparse-land model framework and one of the most fundamental algorithms used for image compression, image denoising and image deblurring. In both algorithms, the basic idea is to estimate the dictionary $\mathbf{D}$ and the sparse coefficient matrix $\mathbf{\Gamma}$ are updated alternately. While one of them is fixed, the other is updated. This cycle is repeated until the stopping criterion is satisfied.

4.2.1 KSVD Algorithm

Algorithm 4 KSVD [23]

Input : Signal set $\mathbf{X}$, initial dictionary $\mathbf{D}^{(0)}$, target sparsity T, number of iterations k

Output : Dictionary $\mathbf{D} \in \mathbb{R}^{m \times n}$ and sparse coefficient matrix $\mathbf{\Gamma} \in \mathbb{R}^{n}$ such that $\mathbf{X} \approx \mathbf{D\Gamma}$

Initialization: Set $D := D^{(0)}$

1 **for** $n = 1 \dots k$ **do**

2 $\quad \forall i : \bar{\gamma}_i := \text{argmin } \|\bar{x}_i - \mathbf{D}\bar{\gamma}_i\|_2 \quad \text{s.t. } \|\bar{\gamma}\|_0 \leq T$

3 $\quad$ **for** $j = 1 \dots n$ **do**

4 $\quad\quad \mathbf{D}_j := \bar{0}$

5 $\quad\quad I := \{\textit{indices of the signals in } \mathbf{X} \textit{ whose representations use } \bar{d}_j\}$

6 $\quad\quad \mathbf{E} := \mathbf{X}_I - \mathbf{D\Gamma}_I$

7 $\quad\quad \{\bar{d}, \bar{g}\} := \text{argmin} \|\mathbf{E} - \bar{d}\bar{g}^T\|_F^2 \quad \text{s.t. } \|\bar{d}\|_2 = 1$

8 $\quad\quad \mathbf{D}_j := \bar{d}$

9 $\quad\quad \mathbf{\Gamma}_{j,I} := \bar{g}^T$

10 $\quad$ **end for**

11 **end for**

The algorithm 4 shows the pseudo-code of the KSVD algorithm [23]. In the line 2, the i^{th} column of the matrix $\mathbf{\Gamma}$ is updated by minimizing the residual under sparsity constraint. This problem can be solved by any greedy algorithms but in this research, the batch-OMP algorithm is used. The most innovative part of the KSVD algorithm is the line 5 and the subsequent lines. The basic idea is to update one particular

column of $\mathbf{D}$ and its corresponding row of $\mathbf{\Gamma}$ while keeping the rest of the matrices maintained:

$$\|\mathbf{X} - \mathbf{D}\mathbf{\Gamma}\|_F^2 = \left\|\mathbf{X} - \sum_{j=1}^{m} \bar{d}_j \bar{\gamma}_j^T\right\|_F^2 \tag{4.9}$$

$$= \left\|\left\{\mathbf{X} - \sum_{k\neq\ell} \bar{d}_k \bar{\gamma}_k^T\right\} - \bar{d}_\ell \bar{\gamma}_\ell^T\right\|_F^2$$

$$= \|\mathbf{E}_\ell - \bar{d}_\ell \bar{\gamma}_\ell^T\|_F^2 \tag{4.10}$$

where $\bar{d}_\ell$ is the ℓ^{th} column of $\mathbf{D}$ and $\bar{\gamma}_\ell^T$ is the ℓ^{th} row of $\mathbf{\Gamma}$. The minimization of the right hand side of Eq.(4.10) seems to easily be done by rank-1 approximation via SVD decomposition while $\mathbf{E}_\ell$ is fixed:

$$\{\bar{d}_\ell, \bar{\gamma}_\ell\} := \text{argmin } \|\mathbf{E}_\ell - \bar{d}_\ell \bar{\gamma}_\ell^T\|_F^2 \quad \text{s.t. } \|\bar{d}_\ell\|_2 = 1 \tag{4.11}$$

However, in fact, a solution of this optimization problem is not the correct solution. Because of the nature of the SVD, this optimization suggests a dense solution for $\bar{\gamma}_\ell$ although it needs to be a sparse vector.

To obtain the sparse solution by SVD, the update step requires a little ingenuity. Instead of using all the signals in $\mathbf{X}$, the update step uses only the columns in $\mathbf{X}$ whose sparse representations use the current atom. Let I denote a set of indices of the signals in $\mathbf{X}$ which use the ℓ^{th} atom of the dictionary $\mathbf{D}$, and $\mathbf{X}_I$ and $\mathbf{\Gamma}_I$ denote the sub-matrix of $\mathbf{X}$ and $\mathbf{\Gamma}$ whose indices are specified by I. Using this notation, Eq.(4.9) can be modified as:

$$\|\mathbf{X}_I - \mathbf{D}\mathbf{\Gamma}_I\|_F^2 = \left\|\mathbf{X}_I - \sum_{j=1}^{m} \bar{d}_j \tilde{\gamma}_j^T\right\|_F^2 \tag{4.12}$$

$$= \left\|\left\{\mathbf{X}_I - \sum_{k\neq\ell} \bar{d}_k \tilde{\gamma}_k^T\right\} - \bar{d}_\ell \tilde{\gamma}_\ell^T\right\|_F^2$$

$$= \|\tilde{\mathbf{E}}_\ell - \bar{d}_\ell \tilde{\gamma}_\ell^T\|_F^2$$

$$= \|\mathbf{E} - \bar{d}\bar{g}^T\|_F^2 \tag{4.13}$$

Note that $\tilde{\gamma}_j^T$ is the j^{th} row of the matrix $\mathbf{\Gamma}_I$. For simplicity, the following notations are used: $\mathbf{E} = \tilde{\mathbf{E}}_\ell = \mathbf{X}_I - \sum_{k \neq \ell} \bar{d}_k \tilde{\gamma}_k^T$, $\bar{d} = \bar{d}_\ell$ and $\bar{\gamma} = \bar{\gamma}_\ell$. Therefore, the minimization of the right hand side of Eq.(4.13) is formulated as [53]:

$$\{\bar{d}, \bar{g}\} := \text{argmin } \|\mathbf{E} - \bar{d}\bar{g}^T\|_F^2 \quad \text{s.t. } \|\bar{d}\|_2 = 1 \tag{4.14}$$

This problem can be solved simply by rank-1 approximation via SVD. Although $\bar{g}^T$ is obtained as a dense solution, the sparsity of the matrix $\mathbf{\Gamma}$ is not ruined since $\bar{g}^T$ is merely a sub-vector of $\bar{\gamma}_j^T$.

There is an alternative method for SVD to implement rank-1 approximation. Rubinstein et.al. [23] suggests a faster iterative process to approximate this computation. When $\bar{d}$ is fixed, Eq.(4.14) is expressed as a following problem, which has a closed-form solution:

$$\bar{g}^T = \text{argmin } \|\mathbf{E} - \bar{d}\bar{g}^T\|_F^2 = \frac{\bar{d}^T\mathbf{E}}{\bar{d}^T\bar{d}} = \bar{d}^T\mathbf{E} \tag{4.15}$$

On the other hand, when $\bar{g}$ is fixed, Eq.(4.14) is expressed as a following problem, which also has a closed-form solution:

$$\bar{d} = \text{argmin } \|\mathbf{E} - \bar{d}\bar{g}^T\|_F^2 = \frac{\mathbf{E}\bar{g}}{\bar{g}^T\bar{g}} \tag{4.16}$$

The iterative computation of Eq.(4.26) and Eq.(4.16) for a few rounds gives an accurate result for Eq.(4.14).

4.2.2 Sparse KSVD Algorithm

Difficulties of Dictionary learning

The dictionary learning algorithm is a powerful tool but it has some major problems by its nature. In the following, two of these problems are introduced.

The first problem is a long computational time. In KSVD for example, the multiplication $\mathbf{D}\mathbf{\Gamma}_I$ by using explicit matrix requires nm operations if the size of the dictionary $\mathbf{D}$ is $n \times m$. This computation is repeated for m iterations, which results

in a long computation time. On the other hand, a structured matrix such as a separable DCT matrix requires less computation time as already discussed in subsection 3.1. Suppose the dictionary $\mathbf{D}$ is separable and expressed as

$$\mathbf{D} = \mathbf{D}_{\text{sep}} \otimes \mathbf{D}_{\text{sep}} \tag{4.17}$$

where $\otimes$ represents a Kronecker product and the size of the separable dictionary $\mathbf{D}_{\text{sep}}$ is $\sqrt{n} \times \sqrt{m}$. This is a property of 2D-DCT matrix. If this property holds, the multiplication $\mathbf{D}\boldsymbol{\Gamma}_I$ is equivalent with the following operation:

$$\mathbf{D}_{\text{sep}}^T \boldsymbol{\Gamma}_I' \mathbf{D}_{\text{sep}} \tag{4.18}$$

where $\boldsymbol{\Gamma}_I'$ is a matrix obtained by reshaping $\boldsymbol{\Gamma}_I$ into a size of $\sqrt{m} \times \sqrt{m}$. This computation requires only $2n\sqrt{m}$ operations. Therefore, if this kind of separable property can be exploited, this problem can be reduced.

The second problem is that the images of the learning data is restricted to a small size and a low dimension. Empirically, the KSVD algorithm can only deal with $n < 1000$ [38] where the size of the dictionary $\mathbf{D}$ is $n \times m$, which means the maximum image size of the learning data is around 30×30. The large size of the dictionary results in a too long computation time and over-fitting problem due to too many free variables. One way to circumvent this problem is to apply the dictionary learning to small image patches instead of a whole image.

Structured dictionary: Double sparsity model

One of the approaches to resolve the difficulties is to assume a certain structure of a dictionary. The simplest model of such a structured dictionary is a double sparsity model. In this model, the dictionary is assumed to be written as follows:

$$\mathbf{D} = \mathbf{D}_0\mathbf{A} \quad \text{with } \forall i,\ \|\bar{a}_i\|_0 = T_0 \ \ll n \tag{4.19}$$

where $\mathbf{D}_0$ is a fixed base-dictionary with a fast deployment such as a separable DCT. Moreover, A is a sparse dictionary representation whose columns $\bar{a}_i$ are all sparse.

This idea is inspired by the fact that a dictionary of an image also looks like an image and hence it must have a sparse representation.

This double sparsity model has a number of benefits. First of all, this model makes the computation faster. Since the deployment of $\mathbf{D}_0$ is fast and multiplication by a sparse matrix $\mathbf{A}$ is fast, the multiplication by $\mathbf{D}$ is also fast. Moreover, the number of free variables of $\mathbf{D}$ is much smaller because of its sparsity. As a result, this model enables to deal with a larger, higher dimension signals.

Algorithm

Sparse KSVD algorithm [19] is a developed form the KSVD algorithm using the double sparsity model. It solves the following problem:

$$\{\hat{\mathbf{A}}, \hat{\mathbf{\Gamma}}\} = \text{argmin } \|\mathbf{X} - \mathbf{D}_0\mathbf{A}\mathbf{\Gamma}\|_F^2 \quad \text{s.t. } \forall i,\ \|\bar{a}_i\|_0 \leq T_0,\ \|\bar{\gamma}_i\|_0 \leq T_1 \tag{4.20}$$

Rubinstein [19] uses this algorithm to learn a dictionary of size 64×100 by 2D-DCT and a dictionary of size 512×1000 by 3D-DCT.

The algorithm 5 shows the overview of the sparse KSVD algorithm. The overall structure is similar to the KSVD algorithm: the iteration between an update of a signal representation $\bar{\gamma}$ and an update of a dictionary representation $\bar{a}$.

Algorithm 5 Sparse KSVD [19]

Input : Signal set $\mathbf{X}$, base dictionary $\mathbf{D}_0$, initial dictionary representation $\mathbf{A}^{(0)}$, target sparsity T, number of iterations k

Output : Sparse dictionary representation $\mathbf{A} \in \mathbb{R}^{m \times n}$ and sparse signal representation $\mathbf{\Gamma} \in \mathbb{R}^{n \times N}$ such that $\mathbf{X} \approx \mathbf{D}_0\mathbf{A}\mathbf{\Gamma}$

Initialization: Set $A := A^{(0)}$

1 **for** $n = 1 \dots k$ **do**

2 $\quad \forall i : \bar{\gamma}_i := \text{argmin } \|\bar{x}_i - \mathbf{D}_0\mathbf{A}\bar{\gamma}_i\|_2 \quad \text{s.t. } \|\bar{\gamma}\|_0 \leq T$

3 $\quad$ **for** $j = 1 \dots m$ **do**

4 $\quad\quad \mathbf{A}_j := \bar{0}$

5 $\quad\quad I := \{\textit{indices of the signals in } \mathbf{X} \textit{ whose representations use } \bar{a}_j\}$

6 $\quad\quad \bar{g} := \mathbf{\Gamma}^T_{j,I}$

7 $\quad\quad \bar{g} := \bar{g}/\|\bar{g}\|_2$

8 $\quad\quad \bar{z} := \mathbf{X}_I\bar{g} - \mathbf{D}_0\mathbf{A}\mathbf{\Gamma}_I\bar{g}$

9 $\quad\quad \bar{a} := \text{argmin } \|\bar{z} - \mathbf{D}_0\bar{a}\|_2 \quad \text{s.t. } \|\bar{a}\|_0 \leq T$

10 $\quad\quad \bar{a} := \bar{a}/\|\mathbf{D}_0\bar{a}\|_2$

11 $\quad\quad \mathbf{A}_j := \bar{a}$

12 $\quad\quad \Gamma_{j,I} := (\mathbf{X}_I^T\mathbf{D}_0\bar{a} - (\mathbf{D}_0\mathbf{A}\mathbf{\Gamma}_I)^T\mathbf{D}_0\bar{a})^T$

13 $\quad$ **end for**

14 **end for**

In the line 2 of the sparse KSVD, the i^{th} column of the matrix $\mathbf{\Gamma}$ is updated by minimizing the residual under sparsity constraint while $\mathbf{A}$ is fixed. This problem can be solved by any greedy algorithms; in this research, the batch-OMP algorithm is used. In this step, the sparse structure of the dictionary $\mathbf{D} = \mathbf{D}_0\mathbf{A}$ speeds up the computation.

In the line 3 and the subsequent lines, the update of the matrix $\mathbf{A}$ is computed after the update of $\mathbf{\Gamma}$. Similarly to the KSVD algorithm, the update uses only the columns in $\mathbf{X}$ whose sparse representations use the currently updated column $\bar{a}_\ell$.

Let I denote a set of indices of the signals in $\mathbf{X}$ which use the column $\bar{a}_\ell$, then the representation error can be expressed as follows:

$$\|\mathbf{X}_I - \mathbf{D}_0\mathbf{A}\mathbf{\Gamma}_I\|_F^2 = \left\|\mathbf{X}_I - \mathbf{D}_0 \sum_{i=1}^{m} \bar{a}_i \tilde{\gamma}_i^T\right\|_F^2 \tag{4.21}$$

$$= \left\|\left\{\mathbf{X}_I - \mathbf{D}_0 \sum_{i\neq\ell} \bar{a}_i \tilde{\gamma}_i^T\right\} - \mathbf{D}_0 \bar{a}_\ell \tilde{\gamma}_\ell^T\right\|_F^2 \tag{4.22}$$

$$= \left\|\mathbf{E}_\ell - \mathbf{D}_0 \bar{a}_\ell \tilde{\gamma}_\ell^T\right\|_F^2 \tag{4.23}$$

Note that $\mathbf{X}_I$ and $\mathbf{\Gamma}_I$ denote the sub-matrix of $\mathbf{X}$ and $\mathbf{\Gamma}$ whose indices are specified by I and $\tilde{\gamma}_i^T$ is the i^{th} row of the matrix $\mathbf{\Gamma}_I$. For simplicity, the following notations are used: $\mathbf{E} = \mathbf{E}_\ell = \mathbf{X}_I - \mathbf{D}_0 \sum_{i\neq\ell} \bar{a}_i \tilde{\gamma}_i^T$, $\bar{a} = \bar{a}_\ell$, $\bar{g} = \tilde{\gamma}_\ell^T$. The minimization of the right hand side of Eq.(4.23) is formulated as:

$$\{\bar{a}, \bar{g}\} := \text{argmin } \|\mathbf{E} - \bar{a}\bar{g}^T\|_F^2 \quad \text{s.t. } \|\bar{a}\|_0 \leq T_0 \tag{4.24}$$

Note that there is no sparsity constraint for $\bar{g}$ because it is a sub-vector of a sparse column of $\mathbf{\Gamma}$. This problem can be solved by alternating the update of $\bar{a}$ and the update of $\bar{g}$. First, $\bar{g}$ is updated while $\bar{a}$ is fixed. This problem can be solved by least squares which has a closed-form solution:

$$\bar{g} := \text{argmin } \|\mathbf{E} - \mathbf{D}_0\bar{a}\bar{g}^T\|_F^2 = \frac{\bar{a}^T\mathbf{D}_0^T\mathbf{E}}{\bar{a}^T\mathbf{D}_0^T\mathbf{D}_0\bar{a}} \tag{4.25}$$

Subsequently, $\bar{a}$ is updated while $\bar{g}$ is fixed. This problem is formulated as:

$$\bar{a} := \text{argmin } \|\mathbf{E} - \mathbf{D}_0\bar{a}\bar{g}^T\|_F^2 \quad \text{s.t } \|\bar{a}\|_0 \leq T_0 \tag{4.26}$$

This problem is not straightforward since $\bar{a}$ is multiplied from the left and the right. Rubinstein [19] showed that there is a much easier alternative for this problem under the constraint that $\bar{g}^T\bar{g} = 1$:

$$\bar{a} := \text{argmin } \|\mathbf{E}\bar{g} - \mathbf{D}_0\bar{a}\|_F^2 \quad \text{s.t } \|\bar{a}\|_0 \leq T_0 \tag{4.27}$$

This problem can be easily solved by any greedy algorithms. In this research, the batch-OMP algorithm is used.

4.3 Denoising Simulation

One of many applications of dictionary learning is to remove noise from an image. In this section, such a denoising technique is demonstrated by using KSVD and sparse KSVD algorithms.

4.3.1 Problem Formulation

Suppose that a signal $\bar{x}$ is generated by sparse-land model using a dictionary $\mathbf{D}$ and a sparse coefficient vector $\bar{\gamma}$:

$$\bar{x} = \mathbf{D}\bar{\gamma} \quad \text{with } \|\bar{\gamma}\|_0 \leq T \tag{4.28}$$

This signal is corrupted by additive noise $\bar{v}$:

$$\bar{y} = \mathbf{D}\bar{\gamma} + \bar{v} \quad \text{with } \|\bar{v}\|_2 \leq \varepsilon \tag{4.29}$$

where the noise $\bar{v}$ is often modeled as a Gaussian noise upper-bounded by ε. The best way to remove noise from this signal $\bar{y}$ is to find the best pair of $\mathbf{D}$ and $\bar{\gamma}$ whose representation error is within ε. Therefore, the denoising problem is formulated as a following inverse problem:

$$\{\hat{\gamma}, \hat{\mathbf{D}}\} := \text{argmin } \|\bar{\gamma}\|_0 \quad \text{with } \|\bar{y} - \mathbf{D}\bar{\gamma}\|_2 \leq \varepsilon \tag{4.30}$$

This problem (Eq.(4.30)) looks similar to the dictionary learning problem. However, one major difference is that this problem only has one image as training database which is not sufficient at all. This issue is often fixed by a method called patch-based method, which will be discussed in the next section.

4.3.2 Patch-Based Method

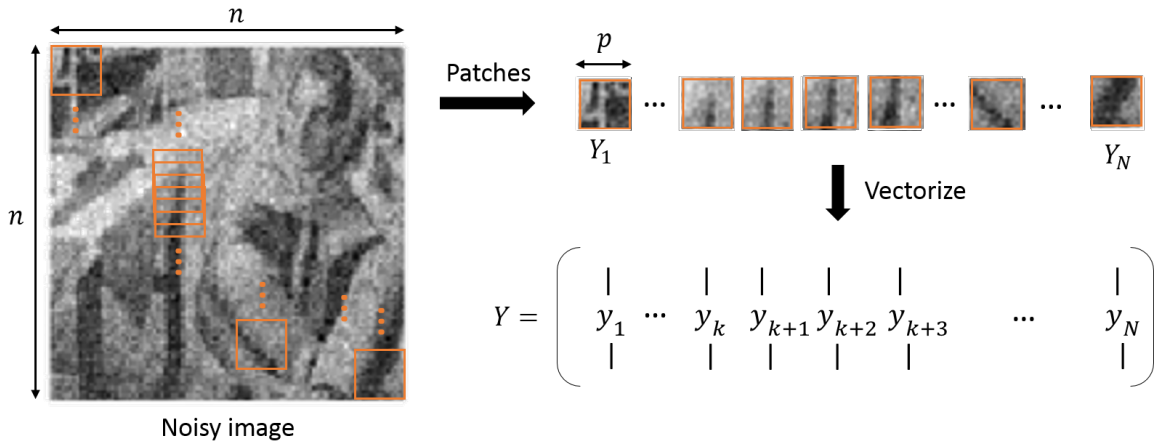


Fig. 4.1.: Patch-based method

The basic idea of a patch-based method is to extract small image patches from a whole image and used them as training database for dictionary learning. This method has multiple advantages. First, it enables to deal with any size of images by choosing a proper patch size. Moreover, it is possible to process each patch image in a parallel way to speed up the computation. Furthermore, it enables to capture local properties of an image which cannot be captured by global-scale operation.

There are some ways to extract patches. The most straightforward way would be to divide the whole image into distinct image patches, apply dictionary learning and merge the result in each patch back to their original locations. This method works well but the result contains some artifacts in the boundary of the patches [52].

A better alternative is to use overlapped patches as shown in Figure 4.1. A patch is first extracted from the upper left corner of the whole image. Then, another patch is extracted from the place one pixel below the first one. This process is repeated until a patch reaches to the lower end. Once it reaches to the end, the patch returns to an upper end of the image but one pixel shifted rightward and the same process is repeated until the patch reaches to the lower right corner. The total number of the

overlapped patches is $(n-p+1)^2$ if the whole image size is $n \times n$ and the patch size is $p \times p$. Each image patch is vectorized and concatenated to form a matrix $\mathbf{Y}$ of size $p^2 \times (n-p+1)^2$. This matrix is used as a training database for dictionary learning algorithm. The denoising problem can be written as follows:

$$\{\hat{\mathbf{\Gamma}}, \hat{\mathbf{D}}\} := \text{argmin } \|\bar{\gamma}_i\|_0 \quad \text{with } \|\mathbf{Y} - \mathbf{D}\mathbf{\Gamma}\|_F \leq \varepsilon \tag{4.31}$$

If the double sparsity model is used, the problem can be written as:

$$\{\hat{\mathbf{\Gamma}}, \hat{\mathbf{A}}\} := \text{argmin } \|\bar{\gamma}_i\|_0 \quad \text{with } \|\mathbf{Y} - \mathbf{D}_0\mathbf{A}\mathbf{\Gamma}\|_F \leq \varepsilon, \ \forall i, \ \|\bar{a}_i\|_0 \leq T_0 \tag{4.32}$$

After each patch is denoised, these results are placed back into their original places and their values are averaged.

4.3.3 Simulation

Methods

In subsection 3.2.2, three images have been reconstructed by simulating a single-pixel camera (Figure 3.9). Now, these noisy images are denoised by using KSVD and sparse KSVD algorithms based on the patch-based method and their results are compared.

For KSVD algorithm, a patch size is 5×5 and a dictionary size is 25×25. The initial dictionary is a 2D-DCT matrix. The constraint for the average representation error per atom is 5. The number of iterations is 100.

For sparse-KSVD algorithm, the patch size is 5×5 and a dictionary size is 25×100. The base dictionary is a separable 2D-DCT dictionary of size 10×10. The sparsity constraint for the dictionary is 16 and the constraint for the average representation error per atom is 5. The number of iterations is 100.

Results

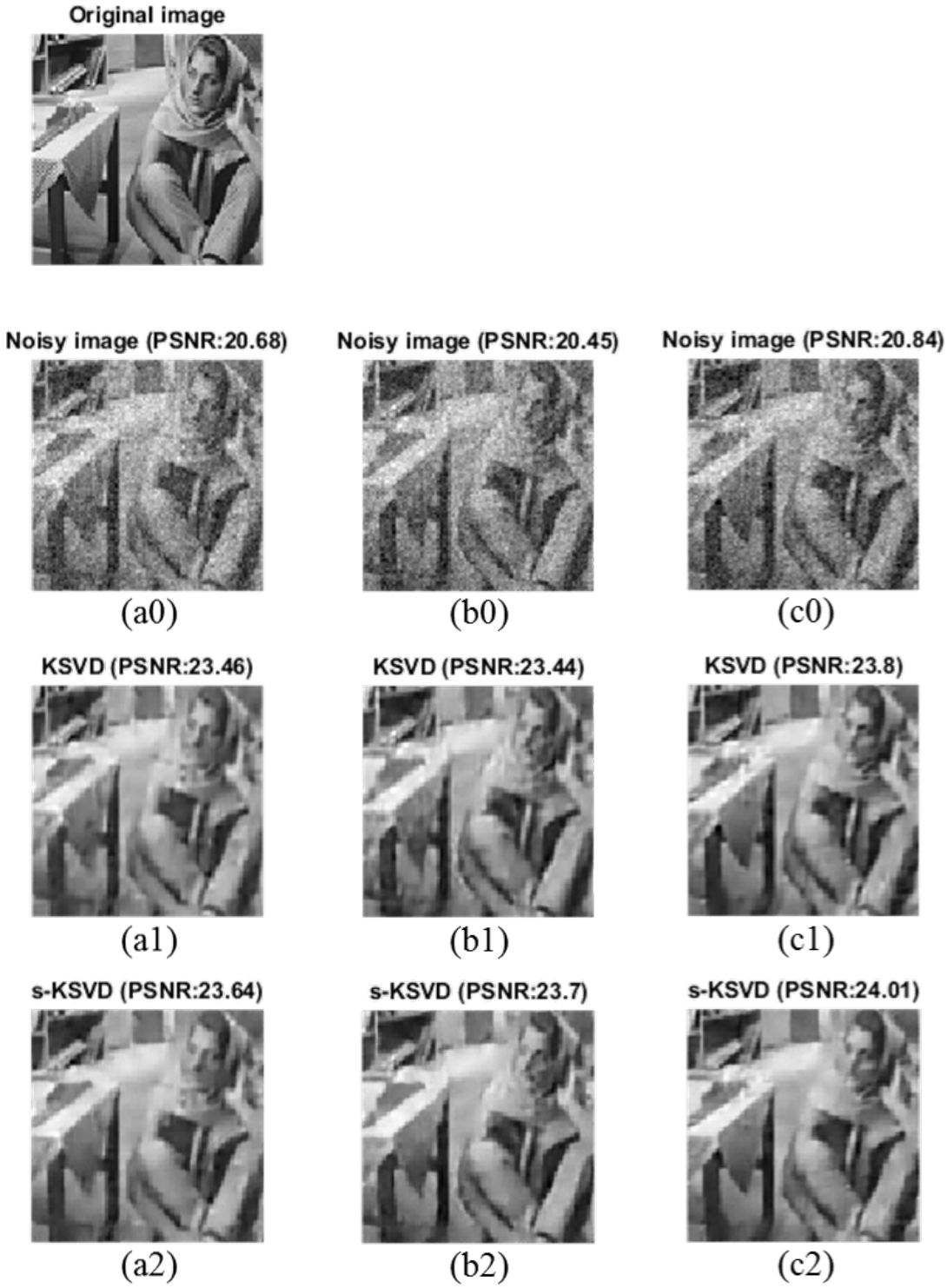


Fig. 4.2.: Comparison of noisy images and denoised images

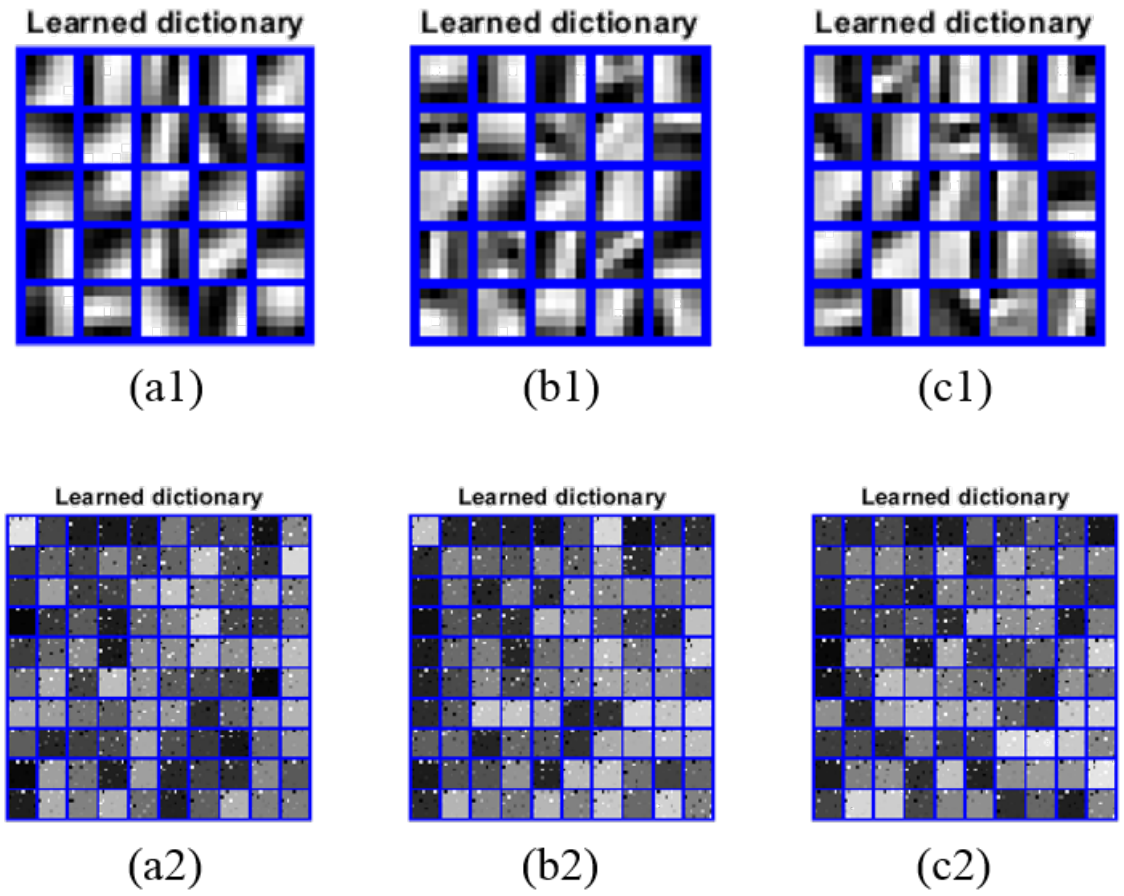


Fig. 4.3.: Comparison of the learned dictionary

Figure 4.2 shows the result for the denoising by KSVD and sparse KSVD algorithm. Figure 4.2 (a0), (b0) and (c0) are the noisy images obtained by Rademacher mirror pattern, Gaussian mirror pattern with 64 minutes exposure and with 640 minutes exposure time in a single-pixel camera simulation in subsection 3.2.2. Figure 4.2 (a1), (b1) and (c1) are the images denoised by KSVD algorithm. Figure 4.2 (a2), (b2) and (c2) are the images denoised by sparse KSVD algorithm.

In both cases, the PSNR values are greatly improved from the noisy images. The sparse KSVD algorithm gives better results than that of the KSVD algorithm by around 0.18-0.2 dB in all three cases. However, the biggest advantage of the sparse

KSVD algorithm is its fast computation time. For the KSVD algorithm, it takes around 39 seconds on average to complete the process for one image. On the other hand, for the sparse KSVD algorithm, the computation time is only 5.6 seconds on average. The simulations have been performed on a Dell Precision Tower 5810 computer equipped with an Intel Xeon E5-1650 and 16 GB of RAM running a 64-bit version of Windows 7.

Figure 4.3 (a1), (b1) and (c1) are the dictionaries $\mathbf{D}$ obtained by KSVD and Figure 4.3 (a2), (b2) and (c2) are the sparse dictionary representations $\mathbf{A}$ obtained by sparse KSVD after 100 iterations. Each column of the dictionary is reshaped and displayed as an image. For example, in the sparse KSVD algorithm, the size of the dictionary representation is $5^2 \times 10^2$. Therefore, in (a2), (b2) and (c2), there are 10×10 squares and the size of each square is 5×5.

As it can be seen, the learned dictionary is optimized into a collection of simple fundamental elements to compose an image. Not surprisingly, Figure 4.3 (a1), (b1) and (c1) share some similar columns since their corresponding images are the same. Figure 4.3 (a2), (b2) and (c2) consist of sparser columns with cardinality 15.

5. CHARACTERIZATION OF SATELLITE BY COMPRESSED SENSING

The main goal of this chapter is to obtain an image of a satellite given its light curve by using the compressed sensing scheme.

5.1 Adapted Light Curve Model and Simulation Overview

5.1.1 Analogy Between Single-Pixel Camera and Light Curve Measurements

Figure 5.1 shows the comparison between (a) a single-pixel camera measurement, (b) an ideal light curve measurement and (c) an attenuated noisy light curve measurement.

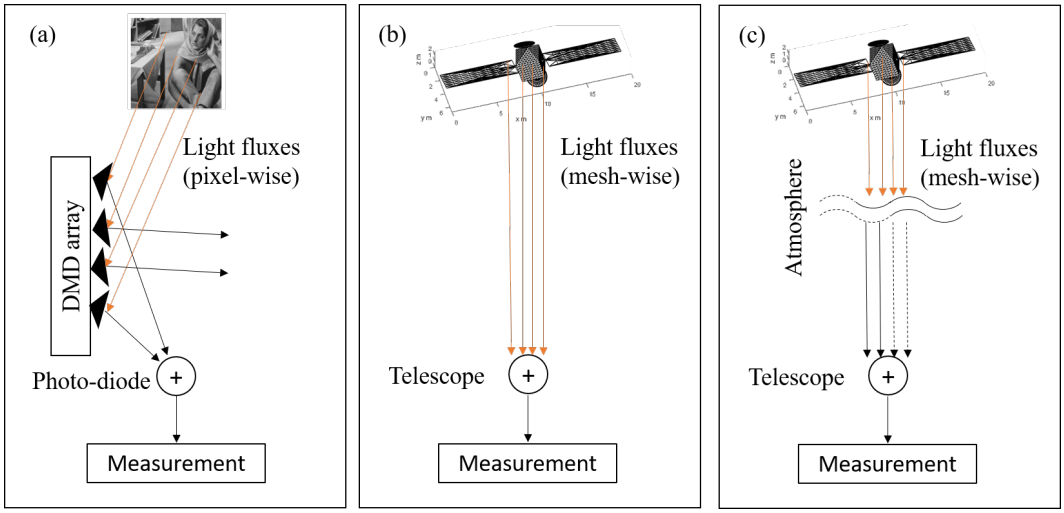


Fig. 5.1.: Comparison of a single-pixel camera and a light curve: (a) Observation of an object by a single-pixel camera, (b) Observation of an object via an ideal light curve, (c) Observation of an object via an attenuated noisy light curve

First, in an observation by a single-pixel camera, the irradiance from an object is multiplied by a random ratio based on the mirror pattern of a DMD array, and their total intensity is measured by a photo-diode as shown in Figure 5.1 (a). The mirror pattern is expressed in a matrix form $\mathbf{\Phi}_k$ and it is vectorized to form a vector $\bar{\Phi}_k$. The k^{th} measurement process is expressed as an inner product between the mirror pattern vector and an image vector $\bar{x}$:

$$y_k = \bar{\Phi}_k^T \ \bar{x} = \begin{bmatrix} \phi_{k1} & \dots & \phi_{kn^2} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n^2} \end{bmatrix} = \sum_{l=1}^{n^2} \phi_{kl} x_l \tag{5.1}$$

This measurement is obtained at m different times using different but known mirror patterns and the measurement vector $\bar{y}$ is obtained.

Similarly, in a light curve observation, light fluxes reflected off a satellite are collected by a sensor. The satellite is modeled as a polygon consisting of small facets (meshes) and the light fluxes are computed in a mesh-wise way:

$$I(t) = I_0 \sum_{i=1}^{n} \frac{A_i}{\pi (r_{topo}(t))^2} \left\{ C_d \ (\bar{V}_i \cdot \bar{N}_i)(\bar{S}_i \cdot \bar{N}_i) + \frac{\tau_i \ C_s \ (r_{Sun}(t))^2}{a_{Sun}^2} \right\} \tag{5.2}$$

$$\tau_i = \begin{cases} 1 & \text{if } \ |\theta_{incoming} - \theta_{outcoming}| \leq 0.5^\circ \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

The Eq.(5.2) can be simplified as an inner product between two vectors:

$$I(t) = \bar{1}^T \bar{\rho}(t) = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \rho_1(t) \\ \vdots \\ \rho_n(t) \end{bmatrix} = \sum_{i=1}^{n} \rho_i \tag{5.4}$$

where ρ_i denotes a light intensity from the i^{th} mesh

$$\rho_i(t) = I_0 \frac{A_i}{\pi (r_{topo}(t))^2} \left\{ C_d \ (\bar{V}_i \cdot \bar{N}_i)(\bar{S}_i \cdot \bar{N}_i) + \frac{\tau_i \ C_s \ (r_{Sun}(t))^2}{a_{Sun}^2} \right\} \tag{5.5}$$

The Eq.(5.4) is a model for an ideal light curve where all the light fluxes are collected without being disturbed as shown in Figure 5.1 (b). However, what can be observed in

reality is an attenuated noisy light curve collapsed by atmosphere as shown in Figure 5.1 (c). The light fluxes from a space object pass through a turbulent atmosphere around the Earth before they reach a ground-based observer. Since the atmosphere is not even due to small temperature variations, the irradiance of light fluxes get fluctuated. Some light fluxes may not be affected much while other fluxes may be weakened severely or reflected off the observer direction. This effect is referred to as scintillation and often modeled as a random process. Therefore, this attenuated noisy light curve may be modeled by simply exchanging the vector $\bar{1}$ in Eq.(5.4) for a Gaussian random vector $\bar{\phi}$:

$$I(t) = \bar{\phi}^T \bar{\rho}(t) = \begin{bmatrix} \phi_1 & \dots & \phi_n \end{bmatrix} \begin{bmatrix} \rho_1(t) \\ \vdots \\ \rho_n(t) \end{bmatrix} = \sum_{i=1}^{n} \phi_i \rho_i \tag{5.6}$$

Interestingly, the Eq.(5.6) is exactly the same form as the Eq.(5.1). Therefore, the attenuated noisy light curve measurement can be regarded as one measurement in a single-pixel camera, and the atmospheric noise can be considered as a mirror pattern in a single-pixel camera, or a sensing matrix in compressed sensing.

However, there are two important aspects to be considered. The first aspect is about a mathematical model of the measurement. The attenuated noisy light curve is simplified to fit into a single-pixel camera model. The validity of this model is discussed in further detail in the next subsection. The second aspect is the availability of the sensing matrix. In a single-pixel camera, the sensing matrix is designed so that it satisfies the RIP condition, and it is used in the reconstruction as well as the measurement. On the other hand, in a light curve measurement, the sensing matrix is unknown because it is impossible to know the atmospheric noise that the light curve is subjected to.

Therefore, the main goal of this research is to bridge a gap between an attenuated noisy light curve and a single-pixel camera under these simplified assumptions, and determine applicability of the compressed sensing scheme to the reconstruction of a resolved satellite image from the light curve.

5.1.2 Adapted Light Curve Model

As introduced in Eq.(5.6) in subsection 5.1.1, this research models the noisy light curve as a product between Gaussian random entries and an irradiance vector:

$$I(t) = \bar{\phi}^T \bar{\rho}(t) = \begin{bmatrix} \phi_1 & \dots & \phi_{n_{\text{mesh}}} \end{bmatrix} \begin{bmatrix} \rho_1(t) \\ \vdots \\ \rho_{n_{\text{mesh}}}(t) \end{bmatrix} = \sum_{i=1}^{n_{\text{mesh}}} \phi_i \rho_i \tag{5.7}$$

where n_{mesh} is a total number of meshes in the 3D model of the satellite. In this research, this light curve model is referred to as the "realistic LC model". In this model, the mesh-wise product is chosen because it is physically correct in terms of the process of the noise generation.

However, this realistic light curve model is not straightforward in terms of the application to compressed sensing. In compressed sensing, an observed satellite is modeled as an image which consists of pixels. Therefore, it is easier to model the noise in a pixel-wise way instead of a mesh-wise way to apply the compressed sensing. Thus, it is assumed that an observed satellite can be modeled as an image. Let $\bar{x}$ denote the image vector, then the alternative model can be expressed as follows:

$$I(t) = \bar{\phi}^T \bar{x}(t) = \begin{bmatrix} \phi_1 & \dots & \phi_{n_{\text{pixel}}} \end{bmatrix} \begin{bmatrix} x_1(t) \\ \vdots \\ x_{n_{\text{pixel}}}(t) \end{bmatrix} = \sum_{i=1}^{n_{\text{pixel}}} \phi_i x_i \tag{5.8}$$

where n_{pixel} is a total number of pixels of the satellite image. In this research this simplified model is referred to as a "simple LC model". This model is an adapted light curve model which is designed to fit into compressed sensing scheme directly.

Turning now to a simulation using this two light curve models, the difference between these two models and a noise-less light curve is discussed. First, a light curve is generated without noise under the same settings as in Figure 2.9 in section 2.1. Second, an attenuated noisy light curve is generated by following the realistic LC model. Third, an attenuated noisy light curve is generated by modeling the satellite

as an image of size 256×256 and following the simple LC model. In the both attenuated noisy light curve models, the noise vector is generated from a Gaussian random variable with mean 0 and variance $1/256^2$ by pseudo-random number generator so that it is consistent with the sensing matrix required by compressed sensing theory. Each time step, the pseudo-random number generator is initialized.

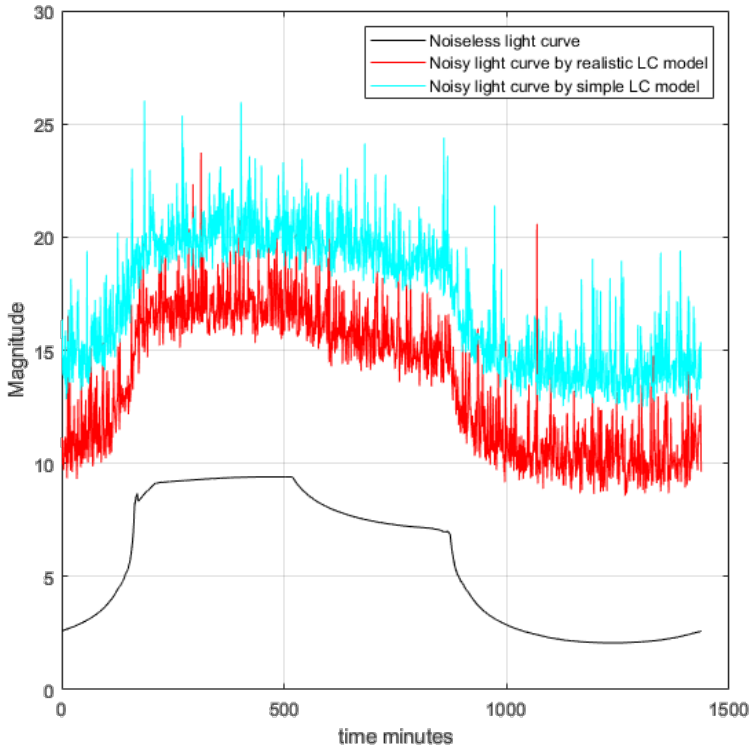


Fig. 5.2.: Comparison of two models for noisy light curve

Figure 5.2 shows the result. The black line is a noise-less light curve which is exactly the same with Figure 2.9. The red line is a realistic LC model and the blue line is a simple LC model. The both attenuated noisy light curves have a similar trend as the noise-less light curve but they are shifted upward, which means they are fainter than noise-less light curve. This characteristic is the same with a light curve observed in a real world so these two models appear to be valid. Another remarkable point is that the simple LC model makes the light curve darker than the realistic

LC model even though the same kind of noise is added. This is simply because the number of pixels (=256^2) are much larger than the number of meshes (=4028) and hence the simple LC model is easier to get affected by noise.

Finally, a procedure to construct a sensing matrix from these two light curve models is considered. In both cases, a satellite needs to be observed m times under the same geometry with varying atmospheric conditions. Therefore, it is assumed that either the m measurements are done in a short time or over a moderate time scale with the satellite being in a geostationary orbit with a stabilized attitude. These m measurements are concatenated vertically to form a measurement vector. For the realistic LC model, the measurement vector is expressed as:

$$\bar{y}_{\text{real}} = \begin{bmatrix} -\phi^{(1)}- \\ \vdots \\ -\phi^{(m)}- \end{bmatrix} \bar{\rho} = \begin{bmatrix} \phi_1^{(1)} & \dots & \phi_{n_{\text{mesh}}}^{(1)} \\ \vdots & & \vdots \\ \phi_1^{(m)} & \dots & \phi_{n_{\text{mesh}}}^{(m)} \end{bmatrix} \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{n_{\text{mesh}}} \end{bmatrix} = \mathbf{\Phi}_{\text{mesh}} \bar{\rho} \tag{5.9}$$

Similarly, for the simple LC model, the measurement vector is expressed as:

$$\bar{y}_{\text{simple}} = \begin{bmatrix} -\phi^{(1)}- \\ \vdots \\ -\phi^{(m)}- \end{bmatrix} \bar{x} = \begin{bmatrix} \phi_1^{(1)} & \dots & \phi_{n_{\text{pixel}}}^{(1)} \\ \vdots & & \vdots \\ \phi_1^{(m)} & \dots & \phi_{n_{\text{pixel}}}^{(m)} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n_{\text{pixel}}} \end{bmatrix} = \mathbf{\Phi}_{\text{pixel}} \bar{x} \tag{5.10}$$

Note that ϕ^k is an atmospheric noise vector in the k^{th} measurement.

In Eq.(5.10), the matrix $\mathbf{\Phi}_{\text{pixel}}$ is identical with a sensing matrix in compressed sensing. Therefore, when a simple LC model is used, the target problem is to directly estimate $\mathbf{\Phi}_{\text{pixel}}$ and $\bar{x}$. On the other hand, in Eq.(5.9), the matrix $\mathbf{\Phi}_{\text{mesh}}$ is not identical to a sensing matrix since $\bar{\rho}$ is not an image vector. Therefore, when a realistic LC model is used, the matrix $\mathbf{\Phi}_{\text{mesh}}$ has to be converted to a sensing matrix based on a relationship between a pixel and a mesh before implementing compressed sensing scheme. This will be discussed further in simulation A of section 5.2.

To summarize, this research considers two light curve models: the realistic LC model and the simple LC model. In both cases, an observed satellite is assumed to be observed from the same orientation under the same light conditions but with m

different atmospheric noise settings. These two models are different in the way they take noise into consideration. In the realistic LC model, the noise is applied in a mesh-wise way. Therefore, this model is physically more accurate but an extra effort is needed to obtain a sensing matrix. On the other hand, in the simple LC model, the noise is applied in a pixel-wise way. Therefore, this model is physically less accurate but more straightforward to apply to compressed sensing.

5.1.3 Simulation Overview

In the following, different satellite image reconstruction scenarios from light curve measurements using compressed sensing are shown. Their methodology, advantages and disadvantages as well as their applicability to real world scenarios are discussed. The first two simulations, A and B solve the satellite image reconstruction problem from light curve measurements in the presence of exact knowledge of the sensing matrix. In the simulation A, the realistic LC model is used while in simulation B, the simple LC model is used. In the simulation A, the limitation of the realistic LC model is shown so in the later simulations, only the simple LC model is used. In the next step, it is assumed that the sensing matrix is not known. In the simulation C1 and C2, the dictionary learning approach is used with and without a patch-based method. Through these two simulations, it is shown that a patch-based method seems to be necessary for the success of the dictionary learning in this research. Therefore, the simulation D uses another approach without using dictionary learning in the absence of the knowledge of the sensing matrix. In this simulation, a light curve and images of a reference satellite are assumed to be known. Under these additional assumptions, a sensing matrix and the resolved satellite image is reconstructed by the compressed sensing scheme. The overview of the assumptions, approaches and results of the simulations are shown in Table 5.1.

Table 5.1.: Simulation overview: assumptions, approaches and results

Simulation	Assumptions		Approach	Result
	LC model	Sensing Matrix $\mathbf{\Phi}$		
A	realistic LC	Given	CS	Success
B	simple LC	Given	CS	Success
C1	simple LC	Not Given	DL with patch	Success
C2	simple LC	Not Given	DL without patch	Failure
D1,D2,D3	simple LC	Not Given	Reference satellite	Success

(Note: LC refers to a light curve, CS refers to compressed sensing and DL refers to dictionary learning.)

5.2 Simulation A: Realistic Light Curve and Φ Known

5.2.1 Simulation

In this simulation, the satellite light curve is modeled as a realistic LC model. Assuming that the noise matrix and the correspondence relationship between the meshes and pixels are perfectly known, an image of the observed satellite is estimated. The summary of this simulation settings is shown in Table 5.2.

Table 5.2.: Problem settings of the simulation A

Condition	Symbol	Size	Description	Assumption
Given	$\bar{y}$	$m \times 1$	realistic LC model	$\bar{y} = \mathbf{\Phi}_{\text{real}}\bar{\rho}$
Given	$\mathbf{\Phi}_{\text{real}}$	$m \times n_{\text{mesh}}$	Pixel-wise noise matrix	$\mathbf{\Phi}_{\text{real}} \sim \mathcal{N}(0, 1/n_{\text{pixel}})$
Find	$\bar{x}$	$\text{n}_{\text{pixel}} \times 1$	satellite image vector	constant

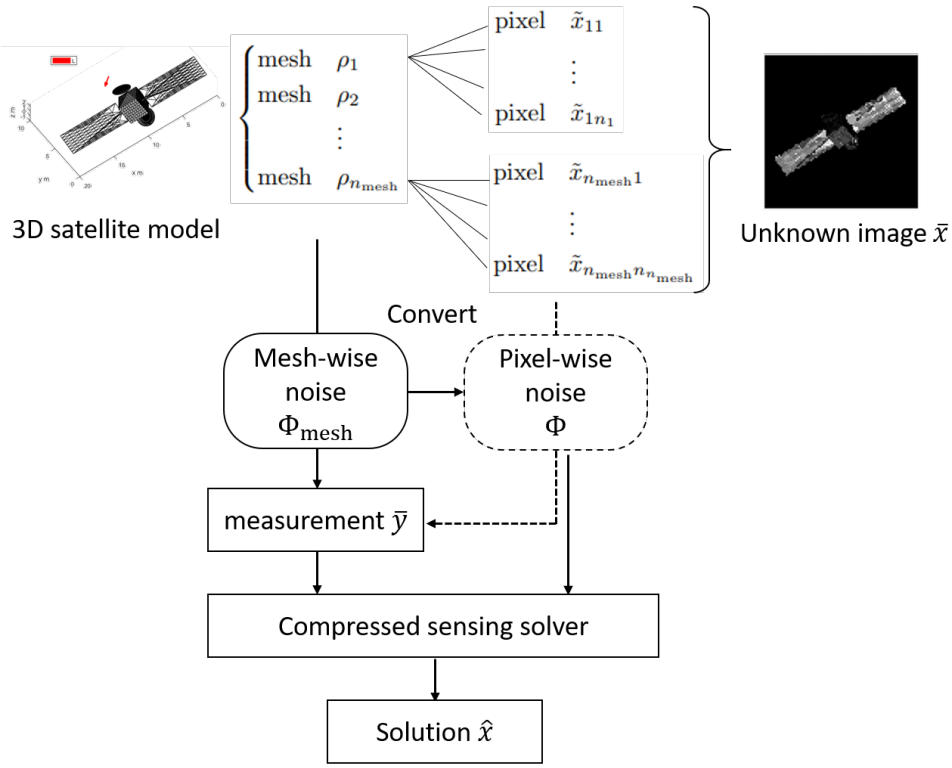


Fig. 5.3.: Flowchart of Simulation A

The procedures for the simulation A is shown in Figure 5.3. First, the light curve intensity $\bar{\rho}$ is corrupted by Gaussian noise matrix $\mathbf{\Phi}_{\text{real}}$ with mean 0 and variance $1/n_{\text{pixel}}$. Note that n_{pixel} is a number of pixels in an image in which the reconstruction is obtained. This matrix $\mathbf{\Phi}_{\text{real}} \in \mathbb{R}^{m \times n_{\text{mesh}}}$ is converted into a sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times n_{\text{pixel}}}$ based on the correspondence relationship between meshes and pixels. Note that m is the number of measurements. Finally, given the sensing matrix $\mathbf{\Phi}$ and the measurement $\bar{y}$, the solution $\hat{x}$ is obtained by solving the compressed sensing problem. First, a sparse coefficient vector $\hat{\gamma}$ is obtained by the minimization of L_0-norm, and subsequently the solution is converted to an image solution $\hat{x}$ by using a discrete cosine transform (DCT) matrix $\mathbf{\Psi}$:

$$\hat{\gamma} = \text{argmin } \|\bar{\gamma}\|_0 \qquad \text{s.t. } \|\bar{y} - \mathbf{\Phi}\mathbf{\Psi}\bar{\gamma}\|_2^2 \leq \varepsilon^2 \tag{5.11}$$

$$\hat{x} = \mathbf{\Psi}\bar{\gamma} \tag{5.12}$$

The approximate problem is also solved by ADMM algorithm (Alternating Direction Method of Multipliers):

$$\hat{\gamma} = \text{argmin } \frac{1}{2}\|\bar{y} - \mathbf{\Phi}\mathbf{\Psi}\bar{\gamma}\|_2^2 + \lambda\|\bar{\gamma}\|_1 \tag{5.13}$$

$$\hat{x} = \mathbf{\Psi}\bar{\gamma} \tag{5.14}$$

Note that λ is a Lagrangian multiplier parameter.

The conversion of a matrix $\mathbf{\Phi}_{\text{real}}$ into a matrix $\mathbf{\Phi}$ is discussed in detail as follows. A mesh-wise measurement of a light curve is expressed as:

$$\bar{y} = \mathbf{\Phi}_{\text{real}}\ \bar{\rho} = \sum_{i=1}^{m} \tilde{\phi}_i^T \bar{\rho} \quad \text{with} \quad \tilde{\phi}_i^T \in \mathbb{R}^{n_{\text{mesh}}} \tag{5.15}$$

where $\tilde{\phi}_i^T$ is the i^{th} row vector of $\mathbf{\Phi}_{\text{real}}$. In order to apply compressed sensing scheme, the measurement needs to be interpreted as a pixel-wise measurement. Therefore, the goal is to find an alternative expression:

$$\bar{y} = \mathbf{\Phi}\bar{x} = \sum_{i=1}^{m} \phi_i^T \bar{x} \quad \text{with} \quad \phi_i^T \in \mathbb{R}^{n_{\text{pixel}}} \tag{5.16}$$

This conversion is done based on a correspondence relationship between meshes and pixels, which will be discussed in the following.

The j^{th} measurement y_j in Eq.(5.15) is a summation of a product between noise values and intensity values:

$$y_j = \tilde{\phi}_j^T \bar{\rho} = \phi_1\rho_1 + \phi_2\rho_2 + \ldots + \phi_{n_{\text{mesh}}}\rho_{n_{\text{mesh}}} \tag{5.17}$$

where $\rho_1, \rho_2, \ldots, \rho_{n_{\text{mesh}}}$ are the intensity of light curve in each mesh, and $\phi_1, \phi_2, \ldots, \phi_{n_{\text{mesh}}}$ are the corresponding noise values. These mesh-wise light intensities are expressed as a summation of pixel-wise light intensities:

$$\rho_1 = \sum_{k=1}^{n_1} \tilde{x}_{1k},\ \rho_2 = \sum_{k=1}^{n_2} \tilde{x}_{2k}, \ldots,\ \rho_{n_{\text{mesh}}} = \sum_{k=1}^{n_{n_{\text{mesh}}}} \tilde{x}_{n_{\text{mesh}}k} \tag{5.18}$$

Substituting Eq.(5.18) into Eq.(5.17), the measurement is expressed in pixel-wise way:

$$y_j = \phi_1 \sum_{k=1}^{n_1} \tilde{x}_{1k} + \phi_2 \sum_{k=1}^{n_2} \tilde{x}_{2k} + \ldots + \phi_{n_{\text{mesh}}} \sum_{k=1}^{n_{n_{\text{mesh}}}} \tilde{x}_{n_{\text{mesh}}k} \tag{5.19}$$

In Eq.(5.19), the pixel intensity values are collected in terms of ϕ's. These terms are sorted in lexicographical order with respect to the linear index of pixels in an image:

$$y_j = \phi'_1 x_1 + \phi'_2 x_2 + \ldots + \phi'_{n_{\text{pixel}}} x_{n_{\text{pixel}}} = \bar{\phi}_k^T \bar{x} \tag{5.20}$$

Eq.(5.20) corresponds to the j^{th} element of the measurement in Eq.(5.16). Therefore, $\bar{\phi}_k^T$ in Eq.(5.20) is the k^{th} row of the sensing matrix $\mathbf{\Phi}_{\text{real}}$.

Therefore, the procedure for converting $\mathbf{\Phi}_{\text{real}}$ into $\mathbf{\Phi}$ can be summarized as follows. First, a mesh-wise noise matrix $\mathbf{\Phi}_{\text{real}}$ is generated from a Gaussian random variable. Each column vector of $\mathbf{\Phi}_{\text{real}}$ corresponds to a mesh, and each mesh corresponds to particular pixels in an image. Suppose the column vector $\bar{\phi}_k$ corresponds to a mesh consisting of pixels $p_1, p_2, ..., p_{n_k}$. Suppose there is an empty matrix $\mathbf{\Phi}$ of size $m \times n_{\text{pixel}}$, then the vector $\bar{\phi}_k$ is copied and saved in the columns of $\mathbf{\Phi}$ which corresponds to $p_1, p_2, ..., p_{n_k}$.

In the simulation, a satellite is assumed to be in geosynchronous orbit and relative position of the observer, the satellite and the Sun is fixed. The satellite is modeled as a 3D mesh model consisting of 4028 meshes as shown in Figure 5.4. The red arrow shows the direction of the Sun. The reconstruction result is obtained as an image of size 100×100. For the OMP algorithm, the error constraint in Eq.(5.11) is set to be $\varepsilon = 0.1$. The ADMM algorithm is implemented 10 times with different values of the parameter λ. These values are chosen from 10 logarithmically spaced points between 10^{-5} and 10^{-1} and multiplied by $\|(\mathbf{\Phi\Psi})^T \bar{y}\|_2$. In both cases, if the reconstruction is successful, an image in Figure 5.5 is obtained.

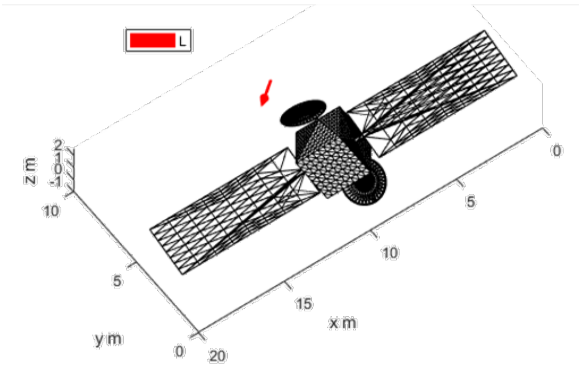


Fig. 5.4.: The 3D satellite model in simulation A

Fig. 5.5.: The desired result in simulation A

5.2.2 Results

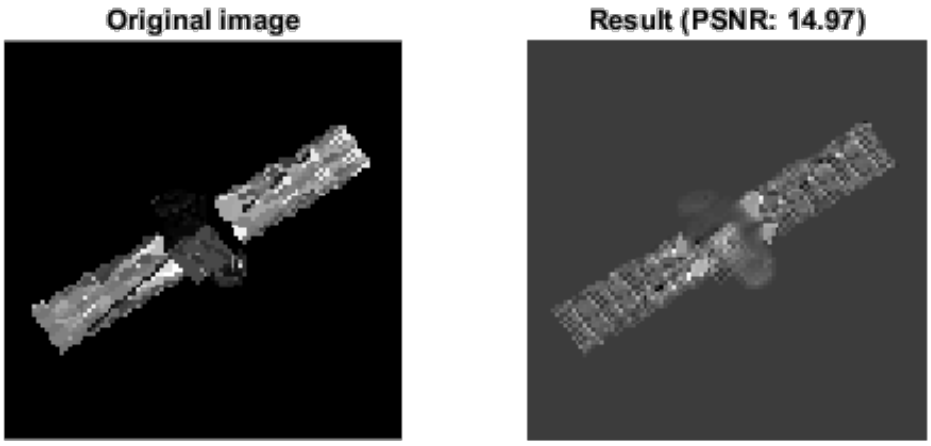


Fig. 5.6.: Reconstructed image in simulation A

In this simulation, a satellite image is reconstructed given its realistic LC and the sensing matrix. The OMP fails to produce a resolved image of the satellite. However, ADMM algorithm gives a resolved image as shown in Figure 5.6. Although some parts of the satellite does not look clear, it is accurate enough to identify the satellite and its sub-components such as bus and panels.

The failure of the OMP is due to the property of the sensing matrix $\mathbf{\Phi}$. The matrix $\mathbf{\Phi}$ has the same multiple columns because there are multiple pixels corresponding to one mesh. As a result, the OMP algorithm cannot find a unique solution since this algorithm chooses the solution based on the inner product between the column of $\mathbf{\Phi}$ and the residual vector.

However, the assumption of this realistic LC model itself is not practical. An image of the observed satellite is assumed to be unknown but at the same time, the correspondence relationship between the meshes and pixels are assumed to be known. In practical cases, this relationship should be unknown as well. If so, there are no ways to reconstruct an image from the light curve.

Therefore, in the following simulations, the realistic LC model is not used. Instead, all the simulations are done using the simple LC model.

5.3 Simulation B: Simple Light Curve and Φ Known

5.3.1 Simulation

In this simulation, a light curve is modeled as a simple LC model. Assuming that a sensing matrix $\mathbf{\Phi}$ is known, an image of the observed satellite is estimated. The summary of the simulation settings is shown in Table 5.3.

Table 5.3.: Problem settings of the simulation B

Condition	Symbol	Size	Description	Assumption
Given	$\bar{y}$	$m \times 1$	simple LC model	$\bar{y} = \mathbf{\Phi}\bar{x}$
Given	$\mathbf{\Phi}$	$m \times n_{\text{pixel}}$	sensing matrix	$\mathbf{\Phi} \sim \mathcal{N}(0, 1/n_{\text{pixel}})$
Find	$\bar{x}$	$n_{\text{pixel}} \times 1$	satellite image vector	constant

The procedures for the simulation B is shown in Figure 5.7. A target image is Figure 5.8 of size 128×128. The pixel value of the image is adjusted so that all the values are between 0 and 255. First, this image is vectorized and expressed as a vector $\bar{x}$. Subsequently, it is measured via sensing matrix assuming a simple LC model:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{5.21}$$

In this measurement process, three different kinds of sensing matrices $\mathbf{\Phi}_1$, $\mathbf{\Phi}_2$ and $\mathbf{\Phi}_3$ are used to see the impact of the number of measurements. All these sensing matrices are generated by Gaussian random variable with mean 0 and variance $1/n_{\text{pixel}}$ where $n_{\text{pixel}} = 128^2$. However, the number of measurements m, or the number of rows of the matrix is $0.81n_{\text{pixel}}$ for $\mathbf{\Phi}_1$, $0.25n_{\text{pixel}}$ for $\mathbf{\Phi}_2$ and $0.09n_{\text{pixel}}$ for $\mathbf{\Phi}_3$.

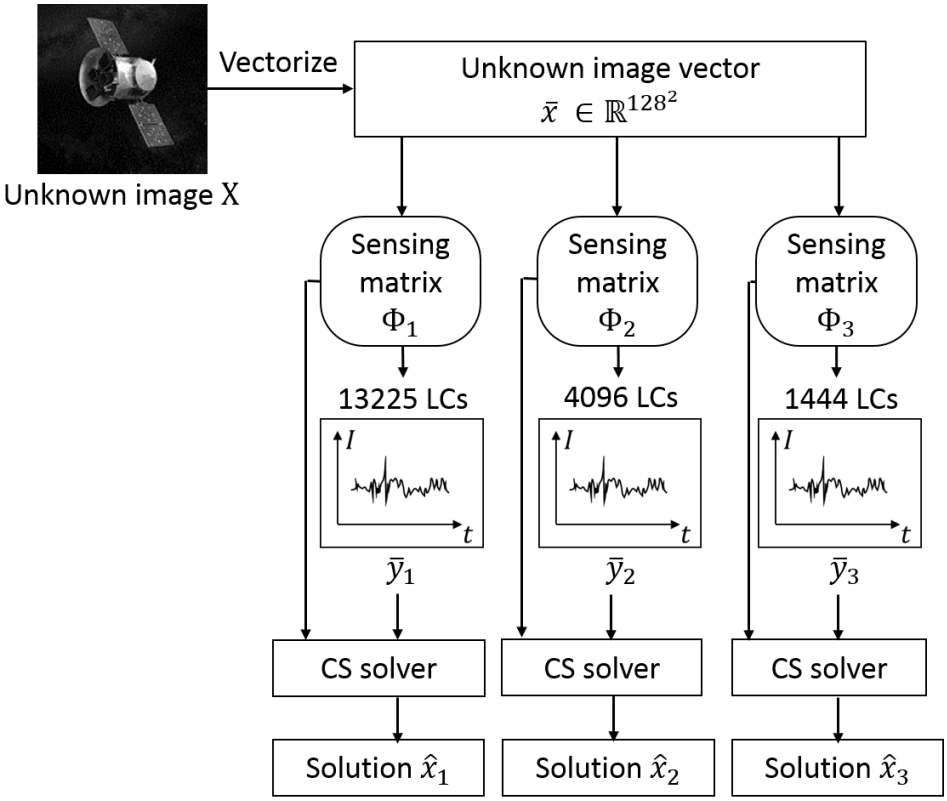


Fig. 5.7.: Flowchart of Simulation B

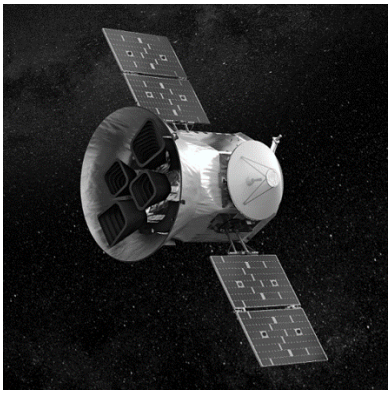


Fig. 5.8.: Satellite image to reconstruct [54]

Second, in each case, the image is reconstructed by using a simple LC and a sensing matrix. Since the sensing matrix is assumed to be known, this problem is exactly the same as the reconstruction process of an image in compressed sensing. A sparse coefficient vector $\hat{\gamma}$ is obtained with either a sparsity constraint or an error constraint:

$$\hat{\gamma} = \text{argmin } \|y - \Phi\Psi\gamma\|_2 \quad \text{s.t. } \|\gamma\|_0 \leq T \text{ or} \tag{5.22}$$

$$\hat{\gamma} = \text{argmin } \|\gamma\|_0 \quad \text{s.t. } \|y - \Phi\Psi\gamma\|_2 \leq \varepsilon \tag{5.23}$$

Subsequently, the sparse coefficient vector $\hat{\gamma}$ is converted to a solution image vector $\hat{x}$ by discrete cosine transform (DCT) matrix:

$$\hat{x} = \Psi\hat{\gamma} \tag{5.24}$$

Theoretically, Eq.(5.22) and Eq.(5.23) are equivalent but empirically, Eq.(5.23) gives a better result. Thus, in this simulation, Eq.(5.23) is solved by using the batch-OMP (*Orthogonal Matching Pursuit*) algorithm [23]. Note that an two-dimensional discrete cosine transform (2D-DCT) matrix is chosen as a sparsifying matrix Ψ. The error constraint is set to be $\varepsilon = 10$.

After each solution is obtained, noise is removed from each result by using a sparse KSVD algorithm. The patch size is 8×8 and the base dictionary is chosen to be a 2D-DCT matrix of size 100×100. Since the noise level differs in each of the three results, different noise thresholds are set. The noise threshold is $\sigma = 10$ for $\mathbf{\Phi}_1$, $\sigma = 15$ for $\mathbf{\Phi}_2$ and $\sigma = 35$ for $\mathbf{\Phi}_3$. The number of iterations is 50. The denoised results are compared based on the PSNR values.

5.3.2 Results

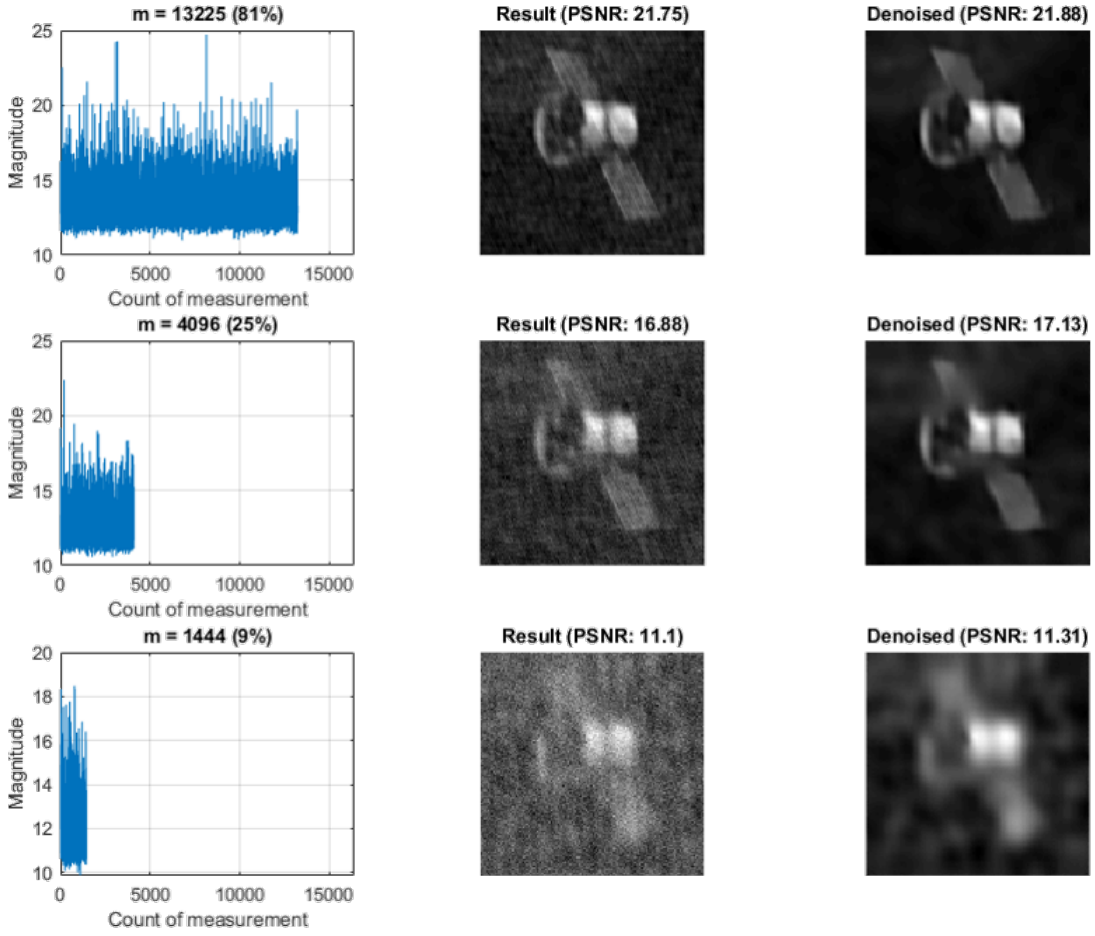


Fig. 5.9.: Reconstruction results with three different number of measurements, m

In this simulation, satellite images are reconstructed given its simple light curve and the sensing matrix with three different number of measurements. The results are shown in Figure 5.9. The first column shows the simple light curve $\bar{y}$ for three different number of measurements ($m = 13225, 4096, 1444$). In the simulation, y is obtained as a vector and it is dimensionless value. However, for the sake of clarity, it is converted to intensity value (W/m^2). In subsection 5.1.2, the intensity value of the noise-less light curve at time $t = 0$ is 1.658×10^{-11} W/m^2. Since the total

pixel value of an image in Figure 5.8 is 494140, the elements of $\bar{y}$ are multiplied by $1.658 \times 10^{-11}/494140$ to have intensity values and its magnitude are computed:

$$\bar{y}_{\text{mag}} = \text{mag}_{Sun} - 2.5 \log_{10}\{1.658 \times 10^{-11}/494140)\bar{y}\} \tag{5.25}$$

where $\text{mag}_{Sun} = -26.74$. This result is plotted as a line graph with index of element in a horizontal axis and the intensity in a vertical axis. Note that some negative elements in $\bar{y}$ are neglected when it is plotted as $\bar{y}_{\text{mag}}$. The values in the parentheses shows the ratio between m and n_{pixel}, which is equivalent with a compression ratio in compressed sensing.

The second column shows the reconstruction results obtained from the first column. The third column shows the results denoised by sparse KSVD algorithm. Each result is shown with its PSNR value.

First of all, Figure 5.9 shows that it is possible to reconstruct an image based on a light curve if a simple LC model is used and the sensing matrix is known. Although the PSNR value of the image deteriorates as the number of measurement decreases, the noise removal process improves the PSNR by $0.1 - 0.2$ dB and makes the image clear enough to identify the satellite shape. In terms of the validity of the assumption that the satellite is observed under the same relative positions between the observer and the Sun, the number of measurement needs to be small. Since the shape of the satellite is still recognizable even when the number of measurement is 4096 in Figure 5.9, the assumption appears to be valid.

5.4 Simulation C: Simple Light Curve and Φ Unknown

In this section, the simple LC model is used but same as before, a satellite image $\bar{x}$ is reconstructed without any prior knowledge about the sensing matrix $\mathbf{\Phi}$.

The absence of the knowledge about the sensing matrix makes the problem extremely difficult to solve. The main reason is a huge number of unknown variables. Reconstruction of an image of size 128×128 requires $\sim 10^4 (= n_{\text{pixel}})$ values of pixels. Suppose this signal is measured by using a sensing matrix of size $0.5n_{\text{pixel}} \times n_{\text{pixel}}$, the matrix has $\sim 10^7$ elements. This means, $\sim 10^4$ variables are needed to specify an image, and $\sim 10^7$ variables are needed to specify a sensing matrix. A problem with too many variables leads to an over-fitting and it is difficult to solve by regular optimization algorithms.

This problem is solved by dictionary learning method. In the first simulation, the benefit of a patch-based method is utilized in dictionary learning. However, this patch-based method is difficult to apply in practical cases. Therefore, in the second simulation, the dictionary learning is implemented without using a patch-based method under simplified problem settings.

5.4.1 Simulation C1: With Patch-Based Method

Patch-based method

In this simulation, a patch-based method is applied to a simple LC model. In the previous simple LC model, a whole satellite image is measured by computing the inner product between the whole image vector $\bar{x}$ and the sensing matrix $\mathbf{\Phi}$:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{5.26}$$

On the other hand, in a patch-based method, it is assumed that only a part of the satellite image can be observed in the form of a square patch of size $p \times p$.

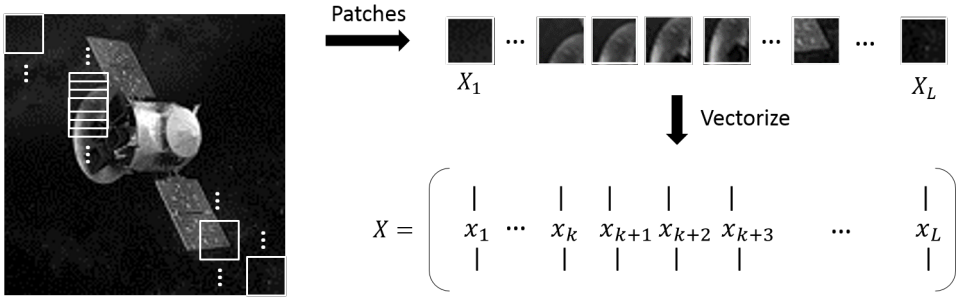


Fig. 5.10.: Process of constructing a matrix X

Figure 5.10 shows a procedure for constructing a image patch matrix $\mathbf{X}$ in a patch-based method. The image patches are cut out so that they are overlapped with each other and span the whole image. The cutout of a patch starts from an upper left of the image and it slides lower by one pixel until the patch reaches to the bottom of the image. Once it reaches to the bottom, a patch returns to the upper of the image which is shifted rightward by one pixel. This process continues until the patch reaches to the lower right of the image. Therefore, suppose that the whole image is of size $n_{\text{edge}} \times n_{\text{edge}}$, the total number of patches being cut out is:

$$N = (n_{\text{edge}} - p + 1)^2 \tag{5.27}$$

Each patch image is measured by smaller sensing matrix $\tilde{\mathbf{\Phi}}$ in the same way as in Eq.(5.26). The patch images are vectorized and the inner product between $\tilde{\mathbf{\Phi}}$ is computed:

$$\left\{ \bar{y}_k = \tilde{\mathbf{\Phi}}_{\text{sub}} \bar{x}_k \right\}_{k=1}^{N} \tag{5.28}$$

These N sets of measurements and image patches can be expressed in the form of a matrix equation:

$$\mathbf{Y} = \tilde{\mathbf{\Phi}} \mathbf{X} \tag{5.29}$$

where $\mathbf{Y}$ has $\bar{y}_k$'s in its columns and $\mathbf{X}$ has $\bar{x}_k$'s in its columns.

This patch-based method may not be a valid assumption. What can be observed in reality is only a summation of all the light fluxes coming from a satellite, and it is impossible to selectively choose a part of the light fluxes coming from a specific facet of the satellite. However, for a while this fact is put aside and benefits of the patch-based method are enjoyed in the expectation that such a technology will be feasible in the future.

The patch-based method has many advantages. The biggest advantage is that it enables us to treat any size images. The size of the sensing matrix is determined by the size of the patches, not by the whole image, so the number of variables as well as the computational cost can be greatly reduced. Moreover, this method helps capture the local characteristic of an image. Furthermore, this method enables us to use a powerful tool called dictionary learning method.

Dictionary learning

A goal is to estimate $\tilde{\mathbf{\Phi}}$ and $\mathbf{X}$ in Eq.(5.29). This problem can be considered as a dictionary learning problem. Assuming that each image patch can be sparsely represented by a 2D-DCT matrix Ψ, the image patches $\mathbf{X}$ can be expressed as follows:

$$\mathbf{X} = \mathbf{\Psi\Gamma} \quad \text{with } \forall i,\ \|\gamma_i\|_0 \leq T \tag{5.30}$$

where γ_i is the i^{th} column vector of a coefficient matrix $\mathbf{\Gamma}$. Substituting Eq.(5.30) into Eq.(5.29), it follows that

$$\mathbf{Y} = \tilde{\mathbf{\Phi}}\mathbf{\Psi\Gamma} = \mathbf{D\Gamma} \tag{5.31}$$

Assume that the sparse expression by Eq.(5.30) is the sparsest possible expression for the matrix $\mathbf{X}$. Then, the target problem is to find the sparsest columns of $\mathbf{\Gamma}$ satisfying $\mathbf{YD\Gamma}$, which is formulated as:

$$\{\hat{\mathbf{D}}, \hat{\gamma}_i\} = \text{argmin } \|\gamma_i\|_0 \quad \text{s.t. } \forall i,\ \|\bar{y}_i - \mathbf{D}\bar{\gamma}_i\|_2 \leq \varepsilon \tag{5.32}$$

This is identical with the dictionary learning problem. Therefore, the problem is solved by one of the dictionary learning algorithms, KSVD algorithm [23].

Simulation

In this simulation, a light curve is modeled as a simple LC model measured from patches of a satellite image. The summary of the simulation settings is shown in Table 5.4.

Table 5.4.: Problem settings of the simulation C1

Condition	Symbol	Size	Description	Assumption
Given	$\mathbf{Y}$	$m \times N$	a set of simple LC's	$\mathbf{Y} = \tilde{\mathbf{\Phi}}\mathbf{X}$
Find	$\tilde{\mathbf{\Phi}}$	$m \times p^2$	sensing matrix	$\tilde{\mathbf{\Phi}} \sim \mathcal{N}(0, 1/p^2)$
Find	$\mathbf{X}$	$p^2 \times N$	satellite image patches	constant

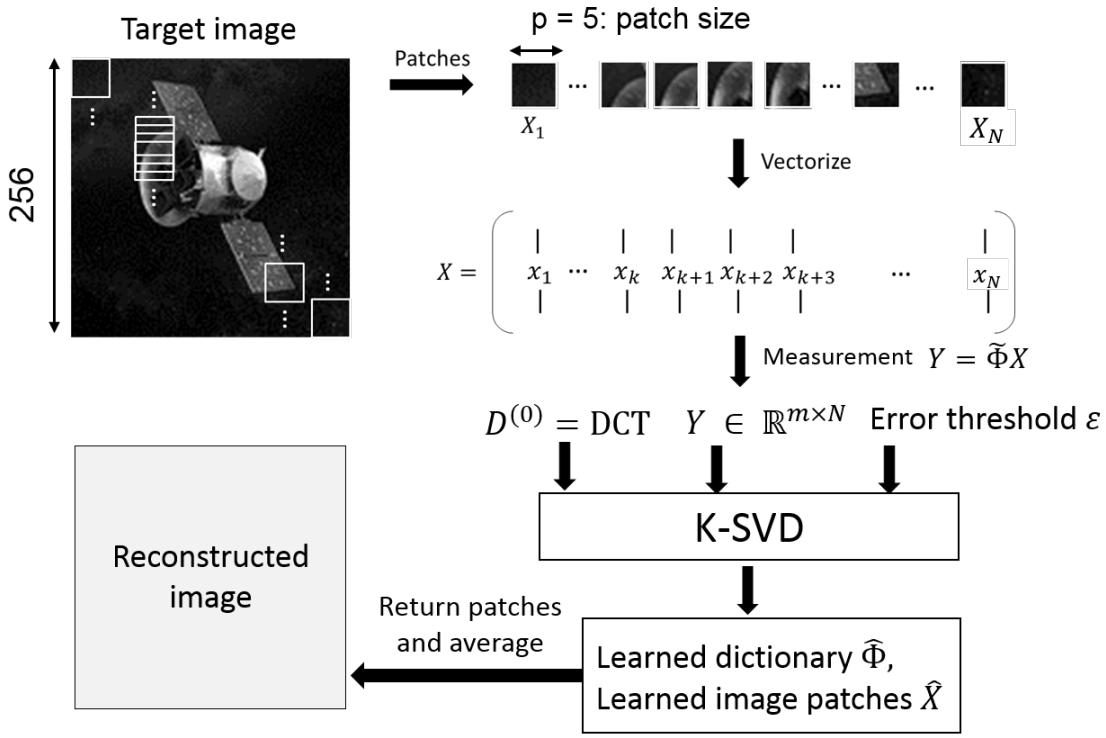


Fig. 5.11.: Flowchart of Simulation C1

The procedure for the simulation C1 is shown in Figure 5.11. A target image is Figure 5.8 of size 256×256. Note that a larger image can be used than that of the simulation B thanks to a patch-based method. This image is then stored as an image patch matrix $\mathbf{X}$. The patch size is 5×5 (hence $p = 5$). It is measured by a sensing matrix $\tilde{\mathbf{\Phi}}$ assuming a simple LC model and a patch-wise light curve $\mathbf{Y}$ is obtained as mentioned earlier. The number of measurements in each patch is $m = 0.64p^2$.

Second, using $\mathbf{Y}$ as an input, the $\mathbf{X}$ and $\tilde{\mathbf{\Phi}}$ are estimated by dictionary learning. First, Eq.(5.32) is solved by KSVD algorithm. The error constraint is set to be $\varepsilon = 0.1$, and the iteration is repeated for 20 times. A 2D-DCT matrix is used as the initial dictionary $\mathbf{\Phi}^{(0)}$. After the coefficient matrix of the patches $\hat{\mathbf{\Gamma}}$ is estimated, the image patch matrix $\hat{\mathbf{X}}$ is computed by Eq.(5.30).

Finally, the whole image solution is obtained. Each column of the solution $\hat{\mathbf{X}}$ is reshaped into an image patch of size 5×5. Subsequently, they are placed back into their original places. Since these patches are overlapping with each other, the average value of the patches is computed.

Results

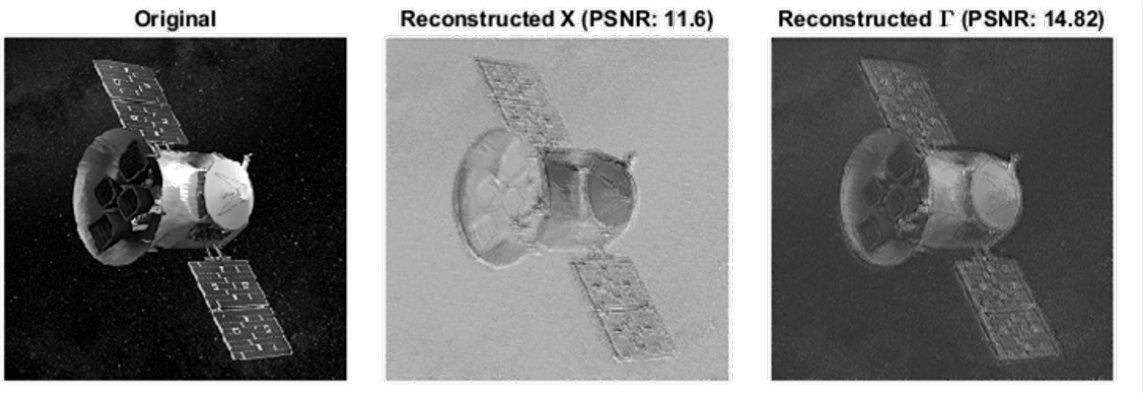


Fig. 5.12.: Reconstruction results by dictionary learning with patch-based method

In this simulation, a satellite image is reconstructed given its simple light curve by a patch-based dictionary learning. Figure 5.12 shows the reconstruction result of the whole satellite image without any prior information about the sensing matrix $\tilde{\mathbf{\Phi}}$. The leftmost image is the original image and the center image is the reconstructed result. This result is obtained by multiplying a DCT matrix $\mathbf{\Psi}$ by $\hat{\mathbf{\Gamma}}$ as in Eq.(5.30). However, the result looks strange. On the other hand, the rightmost image is obtained without multiplying a DCT matrix by $\hat{\mathbf{\Gamma}}$. This result has a larger PSNR value than that of the center one, and it looks closer to the original image. This is probably because image patches are so small that they can be regarded as sparse even without a DCT matrix. In either case, the result is clear enough to recognize the details of the observed satellite.

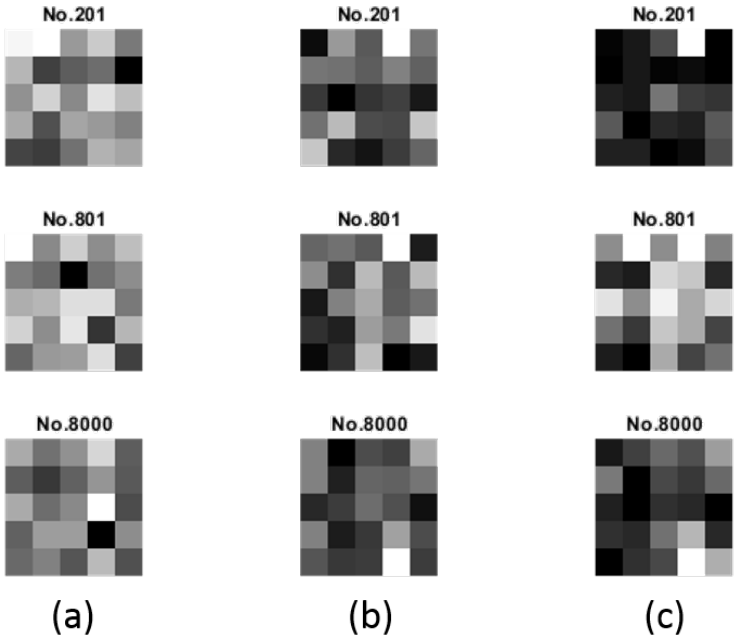


Fig. 5.13.: Comparison of image patches before and after the averaging: (a) Image patches before averaging, (b) Image patches after averaging, (c) Correct image patches

On the other hand, Figure 5.13 shows the reconstruction result of some image patches. The leftmost ones (a) are the patches before placing back to the original places. The leftmost ones (b) are the patches after placing back and averaging. The center ones (c) are the correct patches. It can be seen that the averaging procedure helps the patches get closer to the correct patches. In other words, this dictionary learning method is not accurate without the averaging effect of the patch-based method.

Fig. 5.14.: Comparison of the initial dictionary $\mathbf{D}^{(0)}$, the learned dictionary $\hat{\mathbf{D}}$ and the true dictionary $\mathbf{D}$

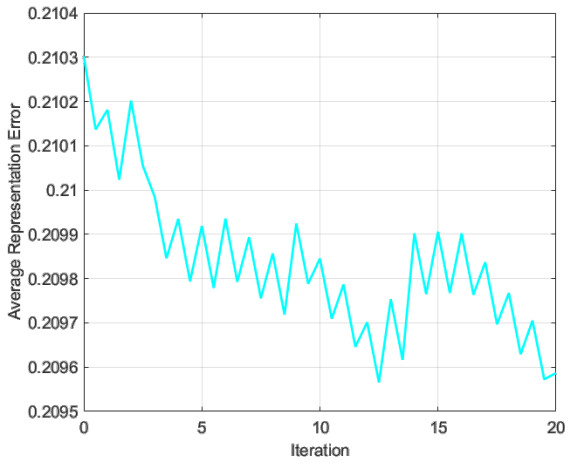


Fig. 5.15.: Representation error during 20 iterations

This is supported by checking the learned dictionary. Figure 5.14 shows the comparison between the initial dictionary, the learned dictionary, and the learned dictionary. It can be seen that the learned dictionary is still somewhat similar to the initial dictionary and it is far from the true dictionary.

One of the possible reasons would be that the iteration has not been converged. However, this possibility is excluded. Figure 5.15 shows the history of the average representation error during the 20 iterations. The error is computed by the following equation:

$$\text{error} = \frac{\|\mathbf{Y} - \hat{\mathbf{D}}\hat{\mathbf{\Gamma}}\|_F}{p^2 \cdot (n_{\text{pixel}} - p + 1)^2} \tag{5.33}$$

where $\|\cdot\|_F$ represents the Frobenius norm. Even when the number of iterations is greater than 20, the error does not get smaller than 0.2096. Therefore, it appears that the dictionary learning has already converged.

In conclusion, the simulation C1 shows the effectiveness of the patch-based dictionary learning approach to characterize a satellite without any prior knowledge about sensing matrix. However, without the patch-based method, the dictionary learning is not accurate enough to estimate the true image and dictionary.

5.4.2 Simulation C2: Without Patch-Based Method

In this simulation, a set of satellite images are measured by sensing matrix to obtain a set of light curve. These light curves are used as a training database and the original images are estimated by dictionary learning without any prior information about the sensing matrix. However, the simulation C1 implied that the accuracy of the dictionary learning is not sufficient enough to estimate a true sensing matrix and a satellite image. Therefore, in this simulation, the problem is simplified and the algorithm is modified to improve the accuracy.

Simplified Problem

Fig. 5.16.: Simplified satellite images for training database

First of all, the problem is simplified. It is assumed that the target image is always observed as a streak image as shown in Figure 5.16. The image size is 16×16 and the total number of images are 10^5. All these images have a common support but with different pixel values. The cardinality of an image is 15, thus these images are regarded as sparse. These images are used as training database for dictionary learning.

This assumption may sound questionable. However, even this simplified image would be helpful for characterization of space objects. For example, it would be possible to discern a difference between a satellite with two solar panels and one with four solar panels. Moreover, if the target satellite is in geosynchronous orbit, it should be observed in a constant location, thus the assumption that all the images have the same support is also reasonable.

Double sparsity model

As discussed in subsection 4.2.2, the double sparsity model helps to improve the accuracy of the dictionary learning while speeding up its computation. In order for this model to be valid, the sensing matrix $\mathbf{\Phi}$ needs to be sparsely represented in terms of a fixed based dictionary $\mathbf{\Phi}_0$:

$$\mathbf{\Phi} = \mathbf{\Phi}_0 \mathbf{A} \quad \text{with } \forall i,\ \|\bar{a}_i\|_0 = T_0 \ll n^2 \tag{5.34}$$

where the size of the image is $n \times n$ ($n = 16$). In order to verify this model for the Gaussian matrix $\sim \mathcal{N}(0, 1/n)$, the sparse representation of the Gaussian matrix is computed by solving the following problem:

$$\mathbf{A} := \text{argmin}\|\bar{a}_i\| \quad \text{s.t. } \|\bar{\phi}_i - \mathbf{\Phi}_0 \bar{a}_i\|_2 \leq \varepsilon \tag{5.35}$$

where $\bar{\phi}_i$ is the i^{th} column vector of the matrix $\mathbf{\Phi}$ and the error constraint is $\varepsilon = 0.1$. The base dictionary $\mathbf{\Phi}_0$ is an overcomplete DCT matrix of size $m \times n^2$ ($m = 64$). This problem is solved by batch-OMP algorithm.

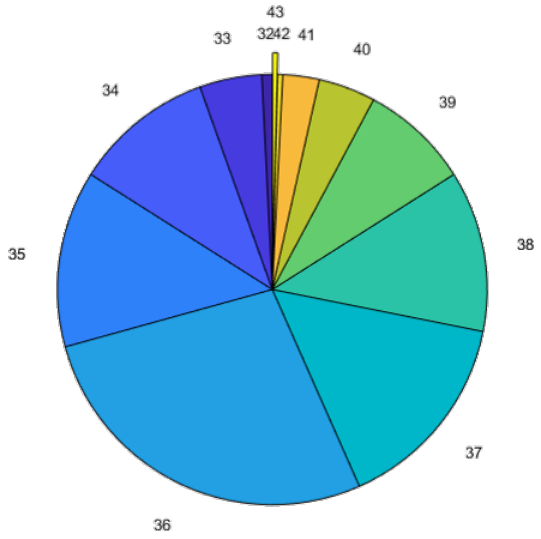


Fig. 5.17.: Cardinality of the representation matrix of the sensing matrix

Figure 5.17 shows the pie chart of the cardinality of the obtained matrix $\mathbf{A}$. The cardinality of each column of the matrix is counted. The label shows the number of cadinality and the angle of the pie shows its percentage. The average cardinality is 36.48 which is only about 14.25% of the number of elements per atom. Therefore, it is safe to say that the double sparsity model holds for this sensing matrix.

Thus, this dictionary learning problem is solved by assuming the double sparsity model. As discussed in 4.2.2, this problem is usually formulated as follows:

$$\{\hat{\mathbf{A}}, \hat{\mathbf{X}}\} := \text{argmin } \|\mathbf{Y} - \mathbf{\Phi}_0\mathbf{A}\mathbf{X}\|_F \quad \text{s.t. } \forall i,\ \|\bar{a}_i\|_0 \leq K_0,\ \|\bar{x}_i\|_0 \leq 15 \tag{5.36}$$

However, this formulation needs a modification in this case. In Eq.(5.36), the sparsity of $\bar{a}_i$ is upper-bounded by K_0 but this results in a matrix $\mathbf{A}$ whose all columns have the cardinality of K_0. This result is not desirable since the correct matrix $\mathbf{A}$ should not have the same cardinality in their columns as in Figure 5.17.

This issue can be circumvented by simply solving the line 9 of the algorithm 5 under the error constraint instead of the sparsity constraint:

$$\bar{a} := \text{argmin}\|\bar{a}\|_0 \quad \text{s.t. } \|\bar{z} - \mathbf{\Phi}_0\bar{a}\|_2 \leq E_0 \tag{5.37}$$

In this case, the cardinality of $\bar{a}$ is not constrained. Thus, it is possible to obtain a matrix A whose columns have variety of cardinalities.

Support-Constrained OMP

The sparse KSVD algorithm is further modified by additional constraint on the solution. Because of the simplification of the problem, all the images have the same support. This fact can be exploited to improve the accuracy of the solution.

In the sparse KSVD algorithm, the update of the representation of the image is done by batch-OMP algorithm in the line 2 in algorithm 5. In this case, the image itself is sparse so the update stage can be written as:

$$\forall i: \ \bar{x}_i := \text{argmin } \|\bar{y}_i - \mathbf{\Phi}_0\mathbf{A}\bar{x}_i\|_2 \quad \text{s.t. } \|\bar{x}_i\|_0 \leq 15 \tag{5.38}$$

This update stage can be modified so that all the columns of $\mathbf{X}$ has the same support.

In the classical OMP algorithm, the support of the solution is chosen by finding the index of the maximum entry of the inner product between a dictionary $\mathbf{\Phi}$ and the residual vector $\bar{r}$ (line 2 in the algorithm 1):

$$\hat{i} := \text{argmax } \|\bar{\phi}_i^T\bar{r}\| \tag{5.39}$$

where $\bar{\phi}_i^T$ is the i^{th} row of $\mathbf{\Phi}$. In each column of the training database, the best supports are chosen by different residual vector $\bar{r}$ (line 5 in algorithm 1):

$$\bar{r} := \bar{y} - \mathbf{\Phi}_I \bar{x}_I \tag{5.40}$$

where I is a chosen support. As a result, the classical OMP obtains a set of images with different supports when the learned dictionary is not accurate enough.

This issue can be fixed by considering a residual matrix $\mathbf{R}$ representing the residual of whole set of solutions:

$$\mathbf{R} := \mathbf{Y} - \mathbf{\Phi}_I \mathbf{X}_I \tag{5.41}$$

Using this residual matrix, a support of the solution is obtained by:

$$\hat{i} := \text{argmax } \|\bar{\phi}_i^T \mathbf{R}\| \tag{5.42}$$

The whole structure of this algorithm is shown in Algorithm 6.

Algorithm 6 Support-constrained OMP

Input : Dictionary $\mathbf{\Phi}$, signal set $\mathbf{Y}$, target sparsity K or target error ε

Output : Sparse representation $\mathbf{X}$ such that $\mathbf{Y} \approx \mathbf{\Phi X}$ and all the columns of $\mathbf{X}$ have the same support

Initialization: Set $I := (), \mathbf{R} := \mathbf{Y}, \mathbf{\Gamma} := \mathbf{0}$

1 **while** *(stopping criterion not met)* **do**

2 $\quad \hat{i} := \text{argmax } \|\bar{\phi}_i^T \mathbf{R}\|$

3 $\quad I := (I, \hat{i})$

4 $\quad \mathbf{X}_I := (\mathbf{D}_I)^+ \mathbf{Y}$

5 $\quad \mathbf{R} := \mathbf{Y} - \mathbf{\Phi}_I \mathbf{X}_I$

6 **end while**

The advantage of the support-constrained OMP can be easily examined through a simple simulation. Let $\mathbf{X}$ denote a matrix of size $16^2 \times 10^5$ whose columns are streak

images with a common support. This matrix is compressed by a known sensing matrix $\mathbf{\Phi}$ to obtain a set of light curves $\mathbf{Y}$

$$\mathbf{Y} = \mathbf{\Phi X} \tag{5.43}$$

Assuming that the matrix $\mathbf{\Phi}$ is known, the image matrix $\mathbf{X}$ is reconstructed by batch-OMP and support-constrained OMP:

$$\hat{\mathbf{X}} = \text{argmin}\|\bar{x}_i\|_0 \quad \text{s.t. } \|\bar{y}_i - \mathbf{\Phi}\bar{x}_i\|_2 \leq \varepsilon \tag{5.44}$$

Note that the error constraint is $\varepsilon = 0.1$.

Figure 5.18 shows the results. As can be seen, there is not a clear difference between the results obtained by the two algorithms. The both algorithms can reconstruct the images with high precision. On the other hand, there is a difference in the computation time. It takes only 2.30 seconds to implement the support-constrained OMP while it takes 4.22 seconds to implement the batch-OMP algorithm.

(a) Original images

(b) Images reconstructed by batch-OMP

(c) Images reconstructed by support-constrained OMP

Fig. 5.18.: Comparison of the results obtained by batch-OMP and support-constrained OMP

In summary, it is expected that the accuracy of the dictionary learning improves by using support-constrained OMP algorithm instead of batch-OMP to solve Eq.(5.38) in the update stage of $\hat{x}_i$ in sparse KSVD algorithm. The support-constrained OMP algorithm efficiently gives a solution under the assumption that the all the solution sets have the same support. Hereinafter, this support-constrained sparse KSVD algorithm is referred to as a modified sparse KSVD algorithm (algorithm 7).

Algorithm 7 Modified sparse KSVD

Input : Signal set $\mathbf{Y}$, base dictionary $\mathbf{\Phi}_0$, initial dictionary representation $\mathbf{A}^{(0)}$, target sparsity of solution T_0, target error of dictionary representation E_0, number of iterations k

Output : Sparse dictionary representation $\mathbf{A} \in \mathbb{R}^{n^2 \times n^2}$ and sparse signal $\mathbf{X} \in \mathbb{R}^{n^2 \times N}$ such that $\mathbf{Y} \approx \mathbf{\Phi}_0 \mathbf{A} \mathbf{X}$

Initialization: Set $A := A^{(0)}$

for $n = 1 \dots k$ **do**
  $\forall i : \bar{x}_i := \text{argmin } \|\bar{y}_i - \mathbf{\Phi}_0 \mathbf{A} \bar{x}_i\|_2 \quad \text{s.t. } \|\bar{x}\|_0 \leq T_0$
  (solved by *support-constrained OMP*)
  for $j = 1 \dots m$ **do**
    $\mathbf{A}_j := \bar{0}$
    $I := \{\textit{indices of the signals in } \mathbf{Y} \textit{ whose representations use } \bar{a}_j\}$
    $\bar{g} := \mathbf{X}_{j,I}^T$
    $\bar{g} := \bar{g} / \|\bar{g}\|_2$
    $\bar{z} := \mathbf{Y}_I \bar{g} - \mathbf{\Phi}_0 \mathbf{A} \mathbf{X}_I \bar{g}$
    $\bar{a} := \text{argmin } \|\bar{a}\|_0 \quad \text{s.t. } \|\bar{z} - \mathbf{\Phi}_0 \bar{a}\|_2 \leq E_0$
    $\bar{a} := \bar{a} / \|\mathbf{D}_0 \bar{a}\|_2$
    $\mathbf{A}_j := \bar{a}$
    $\Gamma_{j,I} := (\mathbf{X}_I^T \mathbf{D}_0 \bar{a} - (\mathbf{D}_0 \mathbf{A} \mathbf{\Gamma}_I)^T \mathbf{D}_0 \bar{a})^T$
  end for
end for

Simulation

In this simulation, a set of light curves $\mathbf{Y}$ is modeled as a simple LC model measured via a sensing matrix $\mathbf{\Phi}$ from a set of satellite images $\mathbf{X}$. These images are reconstructed without any prior knowledge about $\mathbf{\Phi}$. The summary of the simulation settings is shown in Table 5.5.

Table 5.5.: Problem settings of the simulation C2

Condition	Symbol	Size	Description	Assumption
Given	$\mathbf{Y}$	$m \times N$	a set of simple LC's	$\mathbf{Y} = \mathbf{\Phi X}$
Find	$\mathbf{\Phi}$	$m \times n^2$	sensing matrix	$\tilde{\mathbf{\Phi}} \sim \mathcal{N}(0, 1/n^2)$
Find	$\mathbf{X}$	$n^2 \times N$	streak images	common support

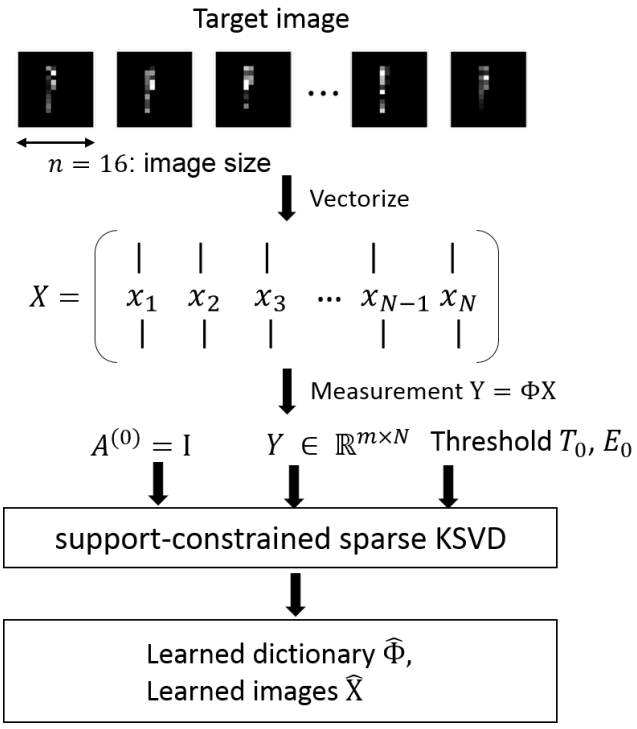


Fig. 5.19.: Flowchart of Simulation C2

The procedure for the simulation C2 is shown in Figure 5.19. The target images are randomly generated streak images of size 16×16 with a common support. Some examples of these images are shown in Figure 5.16. The cardinality of all these images are 15. The total number of the images are 10^5. These images are vectorized and stored as a matrix $\mathbf{X}$ of zize $16^2 \times 10^5$. It is measured by a sensing matrix $\mathbf{\Phi}$ assuming a simple LC model and a set of light curves $\mathbf{Y}$ is obtained. The number of measurements in each image is $m = 0.25 \cdot 16^2 = 64$.

Second, using $\mathbf{Y}$ as an input, the $\mathbf{X}$ and $\mathbf{\Phi}$ are estimated by modified sparse KSVD algorithm. The sparsity constraint of the solution is set to be $T_0 = 15$ and the error constraint of the dictionary representation is set to be $E_0 = 0.1$. The iteration is repeated for 1000 times. The initial dictionary representation is an identity matrix $\mathbf{I}$ of size 256×256.

The same procedure is also implemented by using sparse KSVD algorithm to see the difference. In this case, the sparsity constraint of the dictionary representation is set to be $T_1 = 64$.

Results

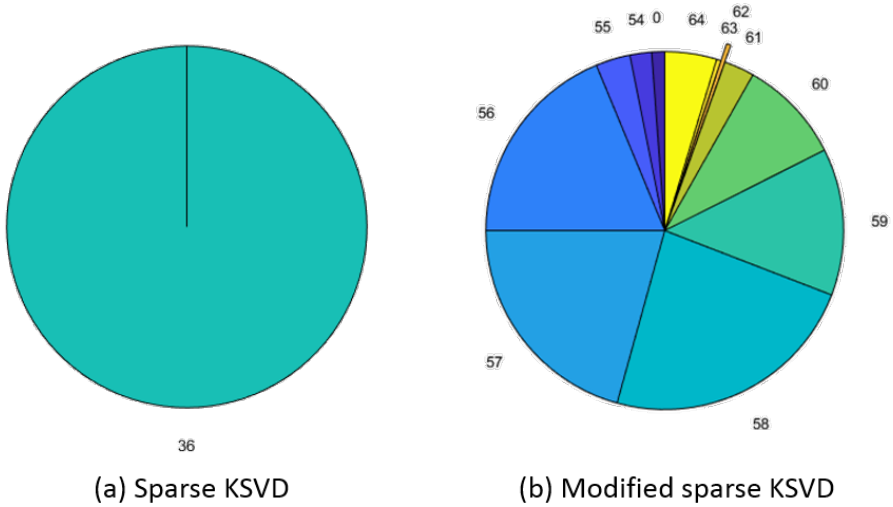


Fig. 5.20.: Cardinality of dictionary representation in two algorithms

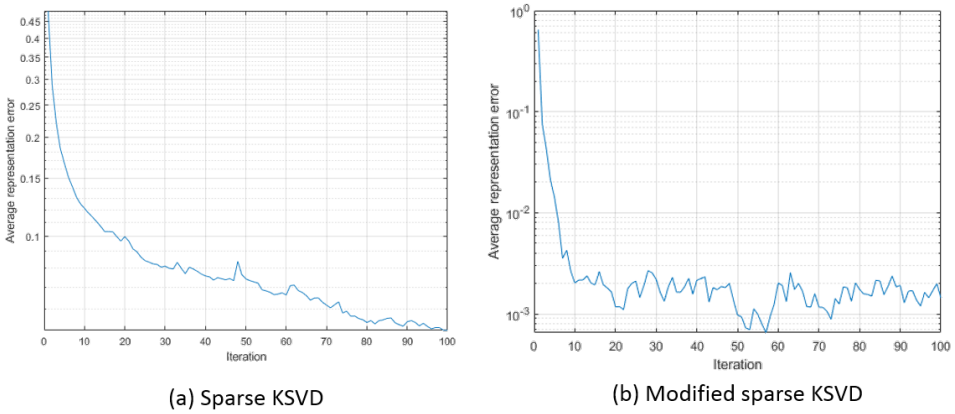


Fig. 5.21.: Changes of representation error in two algorithms

In this simulation, streak images are reconstructed given its simple light curve by sparse KSVD algorithm and modified sparse KSVD algorithm without using a patch-based method. Figure 5.20 and Figure 5.21 show the comparison between the

results obtained by sparse KSVD and modified sparse KSVD after 100 iterations. Figure 5.20 shows the cardinality of each column of the dictionary representation **A**. In the one learned by sparse KSVD, all the columns have the same cardinaliy because of the sparsity constraint. On the other hand, in the one learned by modified sparse KSVD, the columns have a variety of cardinalities ranging from 0 to 64, which captures the structure of the true **A** more accurately. Figure 5.21 shows the changes of the representation error over the number of iterations. While the sparse KSVD decreases the error up to the order of 10^{-1}, the modified sparse KSVD decreases the error up to the order of 10^{-3}. Therefore, clearly, the modified sparse kSVD algorithm has a better performance in dictionary representation as well as in the representation error.

However, even the modified sparse KSVD algorithm does not give a desired result. Figure 5.22 shows the learned images after 1000 iterations. The red boxes show the correct support which should be chosen. Some pixels are correctly chosen but most other pixels are chosen from the incorrect positions. Figure 5.23 focuses more on the chosen support in each iteration. The blue dots represent the position of the chosen pixel in each iteration and the red lines represent the positions of the correct pixels. Ideally, all the 15 blue dots should be on the red lines. However, it never happens during the 1000 iterations and the incorrect pixels (especially 50^{th} and 120^{th} pixels) tend to be chosen frequently. Moreover, in Figure 5.24, the representation error seems to have already converged. Therefore, even if the algorithm is implemented for more iterations, the result would not improve.

There are some possible reasons for these results. One reason would be the low accuracy of the learned dictionary. Although the accuracy is much better than the result obtained by sparse KSVD, the result of the modified KSVD in Figure 5.20 has larger cardinality than the correct dictionary in Figure 5.17. This trend does not change even after 1000 iterations.

In the first place, this algorithm still has a redundancy in the way of learning a dictionary. In this problem, all the images have the same support whose cardinality is

15. Therefore, only the corresponding 15 columns in the dictionary are used to obtain the measurement $\mathbf{Y}$ and the rest of the columns are never used for the calculation of $\mathbf{Y}$. In other words, it is only necessary to learn these 15 columns accurately to obtain the true solution. However, the current algorithm does not take this fact into consideration and it deals with all the columns of the dictionary equally. One of the solutions would be to somehow evaluate the contribution of each column to prune the unnecessary columns, which is similar to the idea of the greedy algorithms.

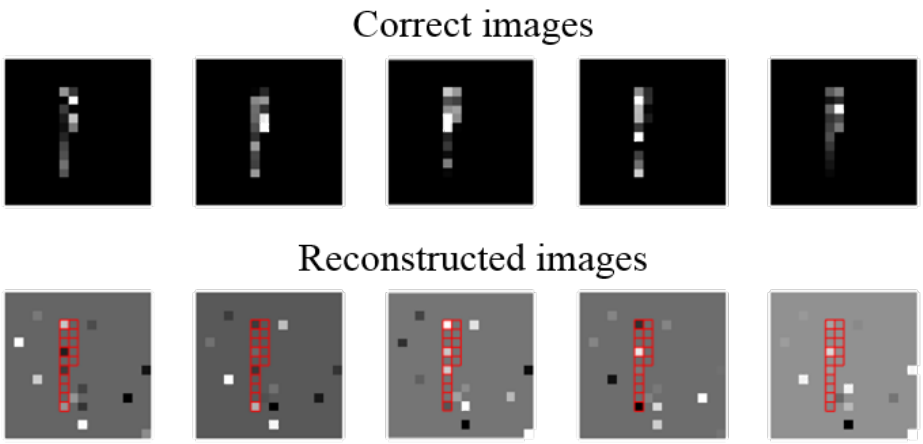


Fig. 5.22.: Images learned after 1000 iterations (by modified sparse KSVD algorithm)

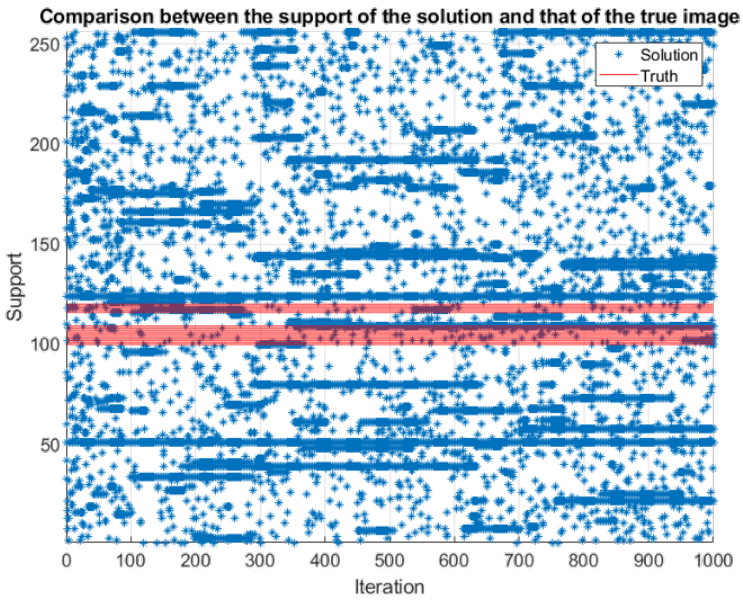


Fig. 5.23.: Support of a solution during 1000 iterations (by modified sparse KSVD algorithm)

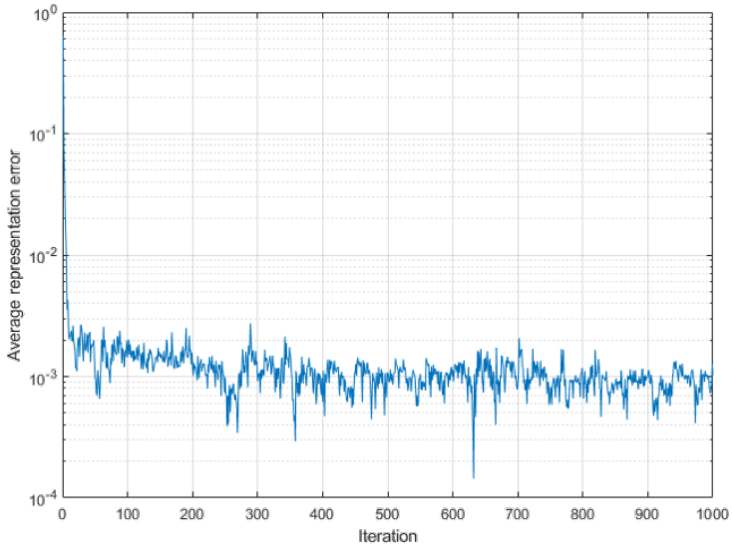


Fig. 5.24.: Representation error during 1000 iterations (by modified sparse KSVD algorithm)

5.5 Simulation D: Simple Light Curve, Reference Satellite and Φ Unknown

In this section as well, a satellite image $\bar{x}$ is reconstructed without any prior knowledge about the sensing matrix. However, one additional assumption is made to substitute the dictionary learning approach. It is assumed that both a light curve and corresponding images of a reference satellite are available.

This method is similar to an idea called a guide star in adaptive optics. The adaptive optics system is a technique to improve the resolution of the astronomical images by sensing and correcting the atmospheric aberrations [25]. In this technique, a reference star or a guide star is observed to estimate the atmospheric turbulence effect. This knowledge enables to adjust a deformable primary mirror to get much sharper images of other stars than images which are obtained without adaptive optics. The guide star can be either a heavenly body or an artificial star such as a sodium laser [26].

Inspired by the idea of a guide star, this section proposes three ways to estimate the sensing matrix by reference satellite observations. In the first simulation, abundant images of the reference satellite are assumed to be available. These images are used to build a full rank matrix, and by using its inversion, a sensing matrix and a target image is estimated. This approach is successful but it has its limitations. In the second simulation, it is assumed that not a sufficient number of images are available to construct a full-rank matrix. Even in such a case, it is shown that a sensing matrix and a target image can be estimated accurately. In the final simulation, the computational cost of the second simulation is lowered by making additional assumptions. It is assumed that a reference satellite is always oriented in a constant direction. It is also assumed that the positions of the target satellite and the reference satellite always overlap in the image.

5.5.1 Simulation D1: Full Rank Matrix Approach

Theory

Assume that there are n_{pixel} $(= n^2)$ images of a reference satellite and their corresponding light curves expressed by simple LC model. The size of each image is $n \times n$. These images are vectorized and concatenated to form a square matrix $\mathbf{X}_{\text{ref}}$ of size $n_{\text{pixel}} \times n_{\text{pixel}}$. Using this matrix, the light curves of the reference satellite images are expressed as:

$$\mathbf{Y}_{\text{ref}} = \mathbf{\Phi}\mathbf{X}_{\text{ref}} \tag{5.45}$$

where $\mathbf{\Phi}$ is a sensing matrix, which is unknown. At the same time, a target satellite image is also observed under the same atmospheric condition. Its light curve is obtained as follows:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{5.46}$$

Suppose the reference image matrix $\mathbf{X}_{\text{ref}}$ is a full-rank matrix, then it has an inverse matrix. Under this assumption, Eq.(5.45) can be simply solved as below:

$$\hat{\mathbf{\Phi}} = \mathbf{Y}_{\text{ref}}\mathbf{X}_{\text{ref}}^{-1} \tag{5.47}$$

Since the sensing matrix $\hat{\mathbf{\Phi}}$ is estimated, Eq.(5.46) can be easily solved by compressed sensing scheme. A sparse coefficient vector $\hat{\gamma}$ is obtained by L_0-norm minimization and subsequently the solution is converted into a solution image vector $\hat{x}$ by using discrete cosine transform (DCT) matrix $\mathbf{\Psi}$:

$$\hat{\gamma} = \text{argmin } \|\bar{\gamma}\|_0 \quad \text{s.t. } \|\bar{y} - \hat{\mathbf{\Phi}}\mathbf{\Psi}\bar{\gamma}\|_2 \leq \varepsilon \tag{5.48}$$

$$\hat{x} = \mathbf{\Psi}\hat{\gamma} \tag{5.49}$$

In this method, two assumptions are made. The first assumption is that both the reference satellite and the target satellite are observed under the same atmospheric conditions or the same sensing matrix. As mentioned earlier, since the similar idea

has been used in adaptive optic technique as well, this assumption seems to be valid. The second assumption is that X_{ref} is a full rank matrix, which means the image vectors of the reference satellite are varied enough to be linearly independent. This assumption holds if the satellite is observed from random directions with random light conditions and noise.

Simulation

In this simulation, a target satellite image and a sensing matrix is estimated by reference satellite images, reference satellite light curves and the target satellite light curve. The summary of the simulation settings is shown in Table 5.6.

Table 5.6.: Problem settings of the simulation D1

Condition	Symbol	Size	Description	Assumption
Given	$\mathbf{X}_{\text{ref}}$	$n_{\text{pixel}} \times n_{\text{pixel}}$	reference image matrix	a full rank matrix
Given	Y_{ref}	$m \times n_{\text{pixel}}$	simple LC of reference satellite	$\mathbf{Y}_{\text{ref}} = \Phi\mathbf{X}_{\text{ref}}$
Given	$\bar{y}$	$m \times 1$	simple LC of target satellite	$\bar{y} = \mathbf{\Phi}\bar{x}$
Find	$\mathbf{\Phi}$	$m \times n_{\text{pixel}}$	sensing matrix	$\mathbf{\Phi} \sim \mathcal{N}(0, 1/n_{\text{pixel}})$
Find	$\bar{x}$	$n_{\text{pixel}} \times 1$	satellite image vector	constant

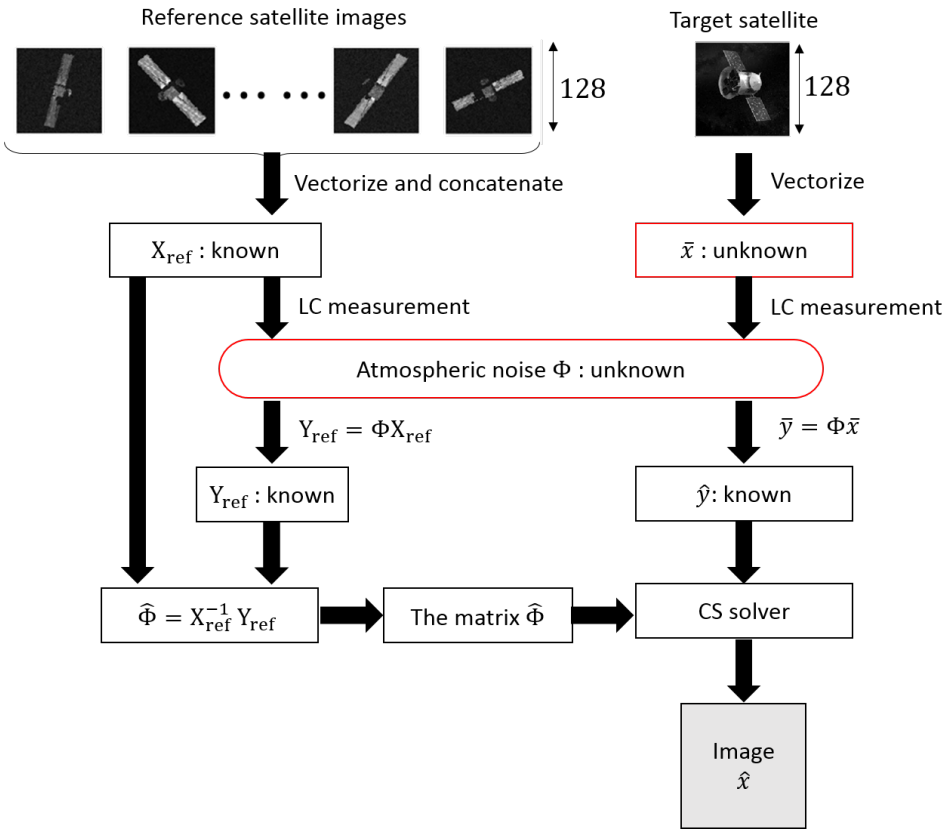


Fig. 5.25.: Flowchart of Simulation D1

The procedure for the simulation D1 is shown in Figure 5.25. A target image is Figure 5.8 of size 128×128. First, reference satellite images of size 128×128 are generated by using the rendering technique in section 2.2. A satellite model in Figure 2.17 is assumed to be in geosynchronous orbit and it is illuminated and observed from random directions. The total number of images are 128^2. Their pixel values are stretched so that all the values are between 0 and 255 and corrupted by Gaussian noise $\sim \mathcal{N}(0, 1)$. Finally, they are vectorized and concatenated to form a square matrix $\mathbf{X}_{\text{ref}}$. This matrix is assumed to be known.

Second, the reference satellite images and the target image are measured by a common sensing matrix $\mathbf{\Phi}$. In both cases, the number of measurements is $m = 0.64 \cdot 128^2 \approx 10486$. A set of the light curves of the reference satellite is expressed

as $\mathbf{Y}_{\text{ref}}$ whose columns correspond to a light curve of each reference image. A light curve of the target satellite is expressed as $\bar{y}$.

Third, the sensing matrix is computed by Eq.(5.47). The inverse of the matrix is computed by using LU decomposition. In order to avoid numerical errors, a matrix $\epsilon\ \mathbf{I}_{n_{\text{pixel}}}$ with a small constant ϵ is added to $\mathbf{X}_{\text{ref}}$ before inverting:

$$\hat{\mathbf{\Phi}} = \mathbf{Y}_{\text{ref}}\ (\mathbf{X}_{\text{ref}} + \epsilon\ \mathbf{I}_{n_{\text{pixel}}})^{-1} \tag{5.50}$$

Note that $\mathbf{I}_{n_{\text{pixel}}}$ is an identity matrix of size $n_{\text{pixel}} \times n_{\text{pixel}}$.

Finally, the target image is estimated by solving Eq.(5.48) based on the estimated sensing matrix $\hat{\mathbf{\Phi}}$ and the light curve $\bar{y}$. The error constraint is 0.1. If the reconstructed image is noisy, the noise is removed by sparse KSVD algorithm. A patch size is 5×5 and a dictionary is 2D-DCT matrix of size 100×100. Sparsity constraint of the dictionary is $T = 6$. The iteration is repeated 50 times.

Results

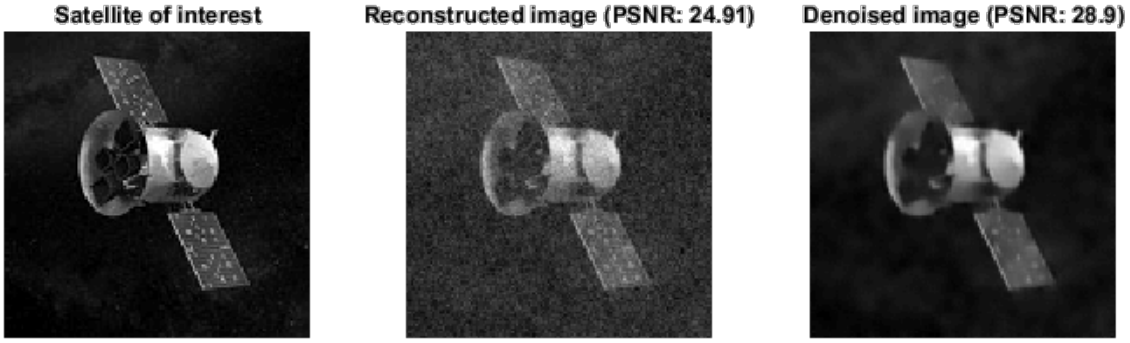


Fig. 5.26.: Reconstruction results of simulation D1

In this simulation, a target satellite image is reconstructed given its light curve and a set of light curves and images of the reference satellite. The images of the reference satellite are assumed to be abundant. Figure 5.26 shows a comparison of the correct image (leftmost), a reconstructed image (center) and a denoised image (rightmost).

As can be seen, even the noisy result clearly shows the shape of the satellite. After the noise has been removed, the image gets even clearer and the result is almost identical with the true image. It takes about 68 minutes to finish the computation in a SID server.

Although this result looks satisfactory, there is still some room for improvement. First of all, this method requires enormous number of measurements. It is necessary to prepare at least n^2 images of the reference satellite if the observed satellite image is of size $n \times n$. Moreover, the construction of the matrix $\mathbf{X}_{\text{ref}}$ is not straightforward. In this simulation, it was straightforward because the synthetic images were corrupted by Gaussian noise, which makes all the images linearly independent with ease. However, this is not the case in actual observations. Just computing the rank of a matrix $\mathbf{X}_{\text{ref}}$ is a difficult task because of its large size. To overcome the limitation of having a large amount of data available and to lower computation times, the simulation D2 and D3 are implemented.

5.5.2 Simulation D2: Rank Deficient Matrix Approach

Theory

The simulation D2 considers a case where the number of images and light curves of a reference satellite is not abundant. Assume that there are only n_{small} ($< n_{\text{pixel}}$) images of a reference satellite and their corresponding light curves expressed by simple LC model. Note that the size of each image is $n \times n$ and the total number of pixels per image is $n_{\text{pixel}} = n^2$. These images are vectorized and concatenated to form a vertical matrix $\mathbf{X}_{\text{ref}} \in \mathbb{R}^{n_{\text{pixel}} \times n_{\text{small}}}$. Using this matrix, the light curves of the reference satellite images are expressed as:

$$\mathbf{Y}_{\text{ref}} = \mathbf{\Phi}\mathbf{X}_{\text{ref}} \tag{5.51}$$

where $\mathbf{\Phi}$ is a sensing matrix, which is unknown. For a later use, both sides of Eq.(5.51) are transposed:

$$\mathbf{Y}_{\text{ref}}^T = \mathbf{X}_{\text{ref}}^T \mathbf{\Phi}^T \tag{5.52}$$

At the same time, a target satellite image is also observed under the same atmospheric condition. Its light curve is obtained as follows:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{5.53}$$

Now the matrix $\mathbf{\Phi}$ needs to be somehow estimated. Since $\mathbf{X}_{\text{ref}}$ is neither a square matrix nor a full-rank matrix, its inverse does not exist in this case. One of the best approaches is to utilize the sparsity of the matrix $\mathbf{\Phi}^T$. First, it is sparsely represented by discrete cosine transform (DCT) matrix $\mathbf{\Psi}$:

$$\mathbf{\Phi}^T = \mathbf{\Psi}\mathbf{A} \quad \text{s.t. } \forall i, \|\bar{a}_i\|_0 \leq T_0 \tag{5.54}$$

where a vector $\bar{a}_i$ is the i^{th} column of the coefficient matrix $\mathbf{A}$. Therefore, now the goal is to estimate the matrix $\mathbf{A}$ exploiting its sparsity property. Substituting Eq.(5.54) into Eq.(5.52), the following expression is obtained:

$$\mathbf{Y}_{\text{ref}}^T = \mathbf{X}_{\text{ref}}^T \mathbf{\Psi}\mathbf{A} \quad \text{s.t. } \forall i, \|\bar{a}_i\|_0 \leq T_0 \tag{5.55}$$

Suppose that Eq.(5.54) is the sparsest possible expression for the matrix $\mathbf{\Phi}_T$, then the columns of the coefficient matrix $\mathbf{A}$ can be obtained by the following optimization problem:

$$\forall i,\ \hat{a}_i = \text{argmin } \|\bar{a}_i\|_0 \quad \text{s.t. } \|\mathbf{Y}^T - \mathbf{X}_{\text{ref}}^T \mathbf{\Psi}\mathbf{A}\|_2 \leq \varepsilon_1 \tag{5.56}$$

Once the matrix $\mathbf{A}$ is estimated, the sensing matrix can be computed as:

$$\hat{\mathbf{\Phi}} = \hat{\mathbf{A}}^T \mathbf{\Psi}^T \tag{5.57}$$

Using the estimated sensing matrix, Eq.(5.53) can be easily solved by compressed sensing scheme. A sparse coefficient vector $\hat{\gamma}$ is obtained by L_0-norm minimization

and subsequently the solution is converted into a solution image vector $\hat{x}$ by using discrete cosine transform (DCT) matrix $\mathbf{\Psi}$:

$$\hat{\gamma} = \text{argmin}\|\bar{\gamma}\|_0 \quad \text{s.t. } \|\bar{y} - \hat{\mathbf{\Phi}}\mathbf{\Psi}\bar{\gamma}\|_2 \leq \varepsilon_2 \tag{5.58}$$

$$\hat{x} = \mathbf{\Psi}\hat{\gamma} \tag{5.59}$$

In this method, the sensing matrix $\mathbf{\Phi}$ is estimated by solving Eq.(5.56) by regarding $\mathbf{X}_{\text{ref}}^T$ as a sensing matrix in compressed sensing. Since the accuracy of the reconstruction highly depends on the sensing matrix in compressed sensing, it is important to choose $\mathbf{X}_{\text{ref}}^T$ properly in this simulation to estimate $\mathbf{\Phi}$ accurately. Therefore, in this simulation, reference satellite images are generated as many as possible and n_{small} images are chosen randomly to construct $\mathbf{X}_{\text{ref}}^T$. This random choice is done several times to construct several candidates for $\mathbf{X}_{\text{ref}}^T$.

There may be some other ways to estimate $\mathbf{\Phi}^T$ in Eq.(5.52). The most straightforward way would be a least squares method:

$$\hat{\mathbf{\Phi}}^T = \text{argmin } \|\mathbf{Y}^T - \mathbf{X}^T\mathbf{\Phi}^T\|_F^2 \tag{5.60}$$

which has a closed-form solution as follows:

$$\hat{\mathbf{\Phi}}^T = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{Y}^T \tag{5.61}$$

However, the least squares cannot give a correct solution to this problem in this case. The detailed reason can be found in chapter 6 of a lecture note by Baraniuk [55]. In short, L_2-norm is not a valid measure to find a sparse solution like this matrix $\mathbf{\Phi}^T$. Without the sparsity constraint on a solution, the minimizer of the L_2-norm of the residual does not necessarily mean the correct solution.

Simulation

In this simulation, a target satellite image and a sensing matrix is estimated by reference satellite images, reference satellite light curves and the target satellite light

curve. Assume that not enough number of reference satellite images are available. The summary of the simulation settings is shown in Table 5.7.

Table 5.7.: Problem settings of the simulation D2

Condition	Symbol	Size	Description	Assumption
Given	$\mathbf{X}_{\text{ref}}$	$n_{\text{pixel}} \times n_{\text{small}}$	reference image matrix	$\text{n}_{\text{small}} < n_{\text{pixel}}$
Given	Y_{ref}	$m \times n_{\text{pixel}}$	simple LC of reference satellite	$\mathbf{Y}_{\text{ref}} = \Phi\mathbf{X}_{\text{ref}}$
Given	$\bar{y}$	$m \times 1$	simple LC of target satellite	$\bar{y} = \mathbf{\Phi}\bar{x}$
Find	$\mathbf{\Phi}$	$m \times n_{\text{pixel}}$	sensing matrix	$\mathbf{\Phi} \sim \mathcal{N}(0, 1/n_{\text{pixel}})$
Find	$\bar{x}$	$n_{\text{pixel}} \times 1$	satellite image vector	constant

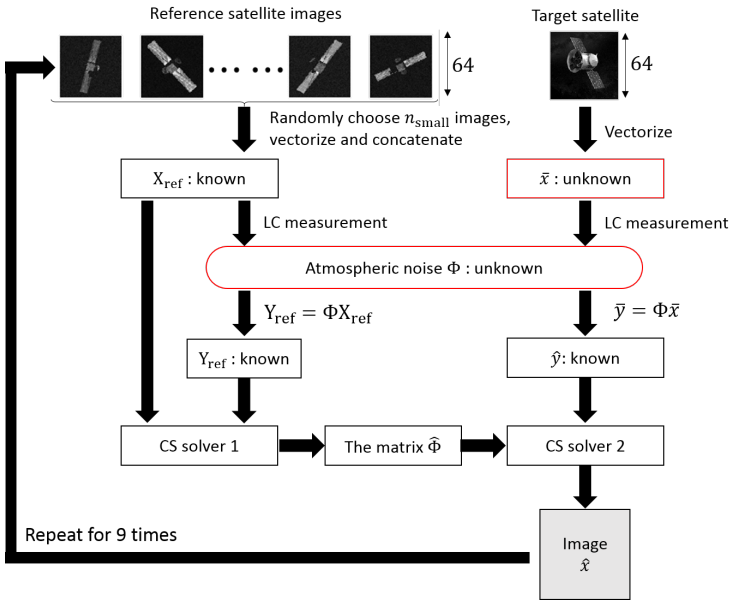


Fig. 5.27.: Flowchart of Simulation D2

The procedure for the simulation D2 is shown in Figure 5.27. A target image is Figure 5.8 of size 64×64. The target image size is smaller than that of the simulation D1 to avoid too much computational cost in the estimation of $\mathbf{\Phi}$.

First, reference satellite images of size 64×64 are generated by using the rendering technique in section 2.2. A satellite model in Figure 2.17 is assumed to be in geosynchronous orbit and it is illuminated and observed from random directions. The total number of images are 8000. Their pixel values are stretched so that all the values are between 0 and 255. Among these images, n_{small} images are randomly chosen, vectorized and concatenated to form a horizontal matrix $\mathbf{X}_{\text{ref}}$. The number of the chosen images is $n_{\text{small}} = 0.8n_{\text{pixel}} \approx 3277$. This matrix is assumed to be known.

Second, the reference satellite images and the target image are measured by a common sensing matrix $\mathbf{\Phi}$. In both cases, the number of measurements is $m = 0.81 \cdot 64^2 \approx 3364$. A set of the light curves of the reference satellite is expressed as $\mathbf{Y}_{\text{ref}}$ whose columns correspond to a light curve of each reference image. A light curve of the target satellite is expressed as $\bar{y}$.

Third, the sensing matrix is computed by solving Eq.(5.56) by batch-OMP algorithm. The error constraint is $\varepsilon = 1$.

Finally, the target image is estimated by solving Eq.(5.58) based on the estimated sensing matrix $\hat{\mathbf{\Phi}}$ and the light curve $\bar{y}$. The error constraint is $\varepsilon = 1$.

This whole procedure is repeated 9 times. In each iteration, different kinds of reference satellite images are chosen at random to construct the matrix $\mathbf{X}_{\text{ref}}$, and the solution $\hat{x}$ is obtained. the best result of the 9 results is chosen. If the image is noisy, the noise is removed by sparse KSVD algorithm with a 2D patch of size 5×5 and a 2D-DCT matrix of size 100×100. Moreover, the 9 results are combined to form a 3D image. The noise of this 3D image is removed by sparse KSVD algorithm with a 3D patch of size $5 \times 5 \times 5$ and a 3D-DCT matrix of size $100 \times 100 \times 100$. This 3D-sparse KSVD algorithm is expected to remove noise better than the 2D-sparse KSVD algorithm.

Results

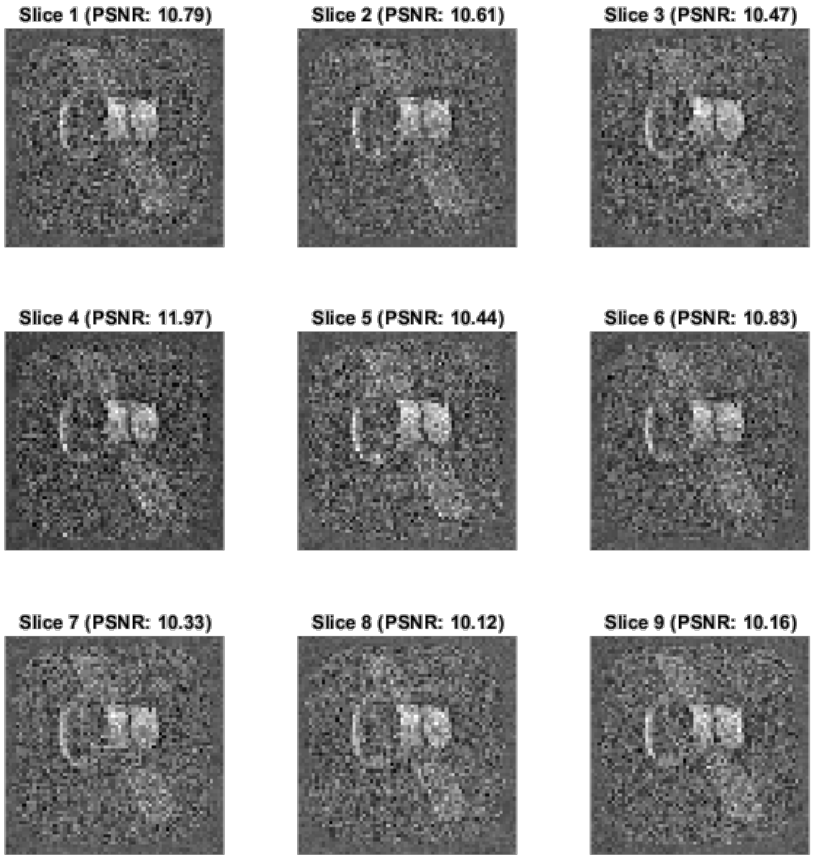


Fig. 5.28.: Noisy results obtained by 9 different $\mathbf{X}_{\text{ref}}$

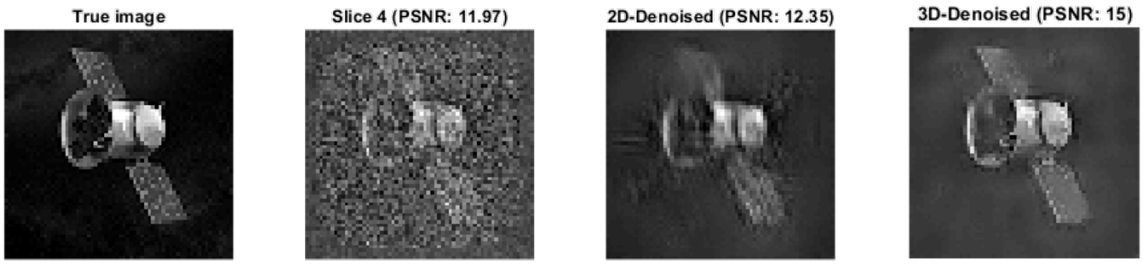


Fig. 5.29.: Comparison between the original image, the best noisy image, 2D-denoised image and 3D-denoised image

In this simulation, a target satellite image and a sensing matrix is estimated by reference satellite images, reference satellite light curves and the target satellite light curve assuming that the reference satellite images are sparse and not adequate to construct a full-rank matrix. Figure 5.28 shows the nine results directly obtained by this method. As expected, the PSNR value of the results are different from each other due to the different sensing matrices $\mathbf{X}_{\text{ref}}$ consisting of different sets of reference satellite images. Overall, the results are noisier than the results obtained by regular compressed sensing because of the ill-designed sensing matrix. For the same reason, the computation time gets longer. It takes about 8 hours to finish the procedure per image. The simulation has been performed in the SID Group server.

Of the nine results, the fourth result has the largest PSNR value in Figure 5.28. Therefore, this image is denoised by 2D-sparse KSVD algorithm. The result is shown in the third image of Figure 5.29. The PSNR value has improved by 0.38 dB but the image is still noisy. Therefore, all the nine results are concatenated to form a 3D image and their noise is removed by 3D-sparse KSVD algorithm. The result is shown in the fourth image of Figure 5.29. The PSNR value has improved by 2.65 dB compared to the 2D-denoised result and the image is clear enough to identify the satellite.

This method has one great advantage over the method proposed in the simulation D1. The restrictions on the reference satellite images are less severe than that of the simulation D1. This method does not require many measurements and the reference satellite images do not have to be linearly independent since the matrix $\mathbf{X}_{\text{ref}}$ does not have to be a full rank matrix. However, due to the ill-designed matrix $\mathbf{X}_{\text{ref}}$, the computational cost is larger and it has a restriction on the size of the image it can deal with. When this simulation is implemented for an image of size 128×128, it takes about 8 days to estimate one image and the result is extremely worse than the one obtained by the simulation D1. Therefore, the simulation D3 attempts to solve this issue by imposing extra constraint on the reference satellite image.

5.5.3 Simulation D3: Faster Full Rank Matrix Approach

Theory

Suppose there are some reference satellite images of size $n \times n$. These images are vectorized and concatenated to form a matrix $\mathbf{X}_{\text{ref}}$. Using this matrix, a set of light curves of the reference satellite can be expressed as:

$$\mathbf{Y}_{\text{ref}} = \mathbf{\Phi}\mathbf{X}_{\text{ref}} \tag{5.62}$$

where $\mathbf{\Phi}$ is a sensing matrix, which is unknown. Similarly, a light curve of a target satellite is also measured:

$$\bar{y} = \mathbf{\Phi}\bar{x} \tag{5.63}$$

Since $\mathbf{\Phi}$ is of size $m \times n_{\text{pixel}}$ (where $n_{\text{pixel}} = n^2$), it is computational expensive to estimate the matrix. However, this issue can be prevented under a certain assumption.

The simulation D3 considers a special case where a reference satellite is always observed from a constant direction. In this case, the reference satellite always occupies a constant position in the images. Therefore, all the vectorized images have the same supports. As a result, all the rows of the matrix $\mathbf{X}_{\text{ref}}$ are zero vectors except for some specific rows. These rows are specified by a set of indices, $I = (i_1, \ldots, i_k)$. Using this set of indices, Eq.(5.62) can be simplified as:

$$\mathbf{Y}_{\text{ref}} = \mathbf{\Phi}(:, I)\ \mathbf{X}_{\text{ref}}(I, :) = \mathbf{\Phi}_{\text{sub}}\mathbf{X}_{\text{sub}} \tag{5.64}$$

Note that $\mathbf{\Phi}(:, I) = \mathbf{\Phi}_{\text{sub}}$ represents a sub-matrix of $\mathbf{\Phi}$ containing the columns indexed by I in the order in which they appear in I. Similarly, $\mathbf{X}_{\text{ref}}(I, :) = \mathbf{X}_{\text{sub}}$ represents a sub-matrix of $\mathbf{X}_{\text{ref}}$ containing the rows indexed by I in their corresponding orders. Generally, the number of non-zero elements k of the reference satellite image is much smaller than the total number of pixels:

$$k \ll n_{\text{pixel}} \tag{5.65}$$

since most pixels in the satellite images are zero because of their black background.

This assumption helps to reduce the computational cost for estimating $\mathbf{\Phi}_{\text{sub}}$. Suppose $\mathbf{X}_{\text{sub}}$ is a square matrix, its size is $k \times k$. Therefore, only k reference satellite images are needed to construct a full-rank matrix $\mathbf{X}_{\text{sub}}$. Moreover, since the images are cropped, it is much easier to collect the images so that they are linearly independent. After constructing a full-rank matrix $\mathbf{X}_{\text{sub}}$, Eq.(5.64) can be simply solved by inverting $\mathbf{X}_{\text{sub}}$:

$$\hat{\mathbf{\Phi}}_{\text{sub}} = \mathbf{Y}_{\text{ref}} \mathbf{X}_{\text{sub}}^{-1} \tag{5.66}$$

However, importantly, this method can only estimate the sub-matrix $\tilde{\mathbf{\Phi}}$ and cannot obtain the whole sensing matrix $\mathbf{\Phi}$. This imposes a certain constraint on a target image which this method can deal with. The support I_{target} of the target satellite image needs to be a subset of the support I:

$$I_{\text{target}} \subseteq I \tag{5.67}$$

In other words, the target satellite and the reference satellite need to overlap in an image. Under this assumption, Eq.(5.63) can be reduced to the following without any loss of information:

$$\bar{y} = \mathbf{\Phi}_{\text{sub}}\ \bar{x}(I) = \mathbf{\Phi}_{\text{sub}}\ \bar{x}_{\text{sub}} \tag{5.68}$$

where $\bar{x}(I) = \bar{x}_{\text{sub}}$ represents a sub-vector of $\bar{x}$ whose elements are indexed by I. Since the sensing matrix $\hat{\mathbf{\Phi}}_{\text{sub}}$ is estimated, Eq.(5.68) can be easily solved by compressed sensing scheme. A sparse coefficient vector $\hat{\gamma}_{\text{sub}}$ is obtained by L_0-norm minimization and subsequently the solution is converted into a solution image vector $\hat{x}_{\text{sub}}$ by using discrete cosine transform (DCT) matrix $\mathbf{\Psi}$ ($\in \mathbb{R}^{k \times k}$):

$$\hat{\gamma}_{\text{sub}} = \text{argmin}\ \|\bar{\gamma}_{\text{sub}}\|_0 \quad \text{s.t.}\ \|\bar{y} - \mathbf{\Phi}_{\text{sub}} \mathbf{\Psi} \bar{\gamma}_{\text{sub}}\|_2 \leq \varepsilon \tag{5.69}$$

$$\hat{x}_{\text{sub}} = \mathbf{\Psi} \bar{\gamma}_{\text{sub}} \tag{5.70}$$

The target image result is obtained by nullifying a vector $\hat{X}$ and substituting $\hat{x}_{\text{sub}}$ in indices I:

$$\hat{x}(I) := \hat{x}_{\text{sub}} \tag{5.71}$$

Simulation

In this simulation, a reference satellite is assumed to be always observed from a constant orientation. Moreover, a target satellite and the reference satellite are assumed to overlap in images. The summary of the simulation settings is shown in Table 5.8.

Table 5.8.: Problem settings of the simulation D3

Condition	Symbol	Size	Description	Assumption
Given	$\mathbf{X}_{\text{sub}}$	$k \times k$	sub-matrix of X_{ref} indexed by I	$\lvert I\rvert = k \ll n_{\text{pixel}}$
Given	$\mathbf{Y}_{\text{ref}}$	$m \times k$	simple LC of reference satellite	$\mathbf{Y}_{\text{ref}} = \mathbf{\Phi}_{\text{sub}}\mathbf{X}_{\text{sub}}$
Given	$\bar{y}$	$m \times 1$	simple LC of target satellite	$\bar{y} = \mathbf{\Phi}_{\text{sub}}\bar{x}_{\text{sub}}$
Find	$\mathbf{\Phi}_{\text{sub}}$	$m \times k$	sub-matrix of sensing matrix	$\mathbf{\Phi}_{\text{sub}} \sim \mathcal{N}(0, 1/k)$
Find	$\bar{x}$	$n_{\text{pixel}} \times 1$	satellite image vector	constant

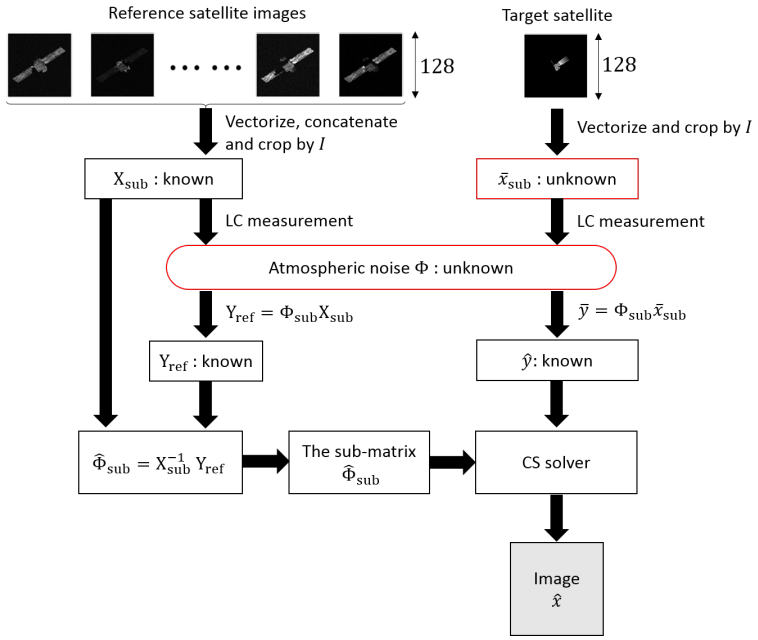


Fig. 5.30.: Flowchart of Simulation D3

The procedure for the simulation D3 is shown in Figure 5.30. Comparison of a reference satellite and a target satellite is shown in Figure 5.31. The leftmost image is one of the examples of the reference satellite image of size 128×128. It is generated by using the rendering technique in section 2.2. A satellite model in Figure 2.17 is assumed to be in geosynchronous orbit and it is observed from a constant direction but with different light conditions. As can be seen in the rightmost image in Figure 5.31, the two satellites overlap in an image so that Eq.(5.67) holds. The pixel values of the reference images are stretched so that all the values are between 0 and 255. These images are cropped by their common support I whose number of elements is $k = 1922$. Subsequently, they are corrupted by Gaussian noise $\sim \mathcal{N}(0, 1)$, vectorized and concatenated to form a square matrix $\mathbf{X}_{\text{sub}}$ of size 1922×1922. The target image is also cropped by I.

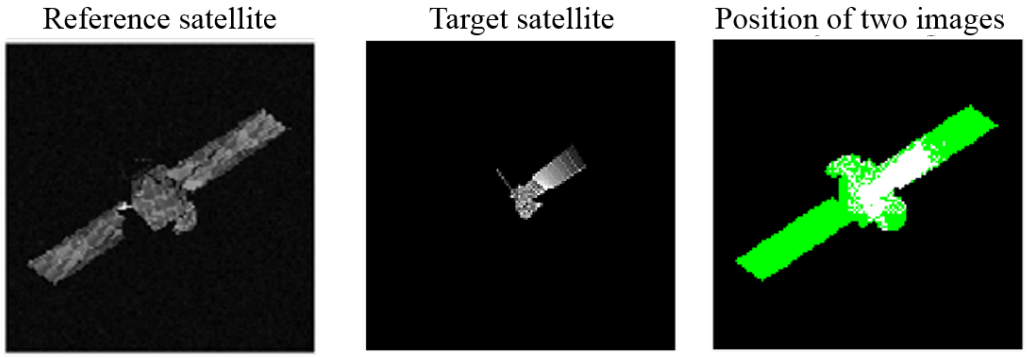


Fig. 5.31.: Reference satellite image, target satellite image and their relationship

Second, the cropped reference satellite images and the cropped target image are measured by a common sensing matrix $\mathbf{\Phi}$. In both cases, the number of measurements is $m = 0.8k \approx 1538$. A set of the light curves of the reference satellite is expressed as $\mathbf{Y}_{\text{ref}}$ whose columns correspond to a light curve of each reference image. A light curve of the target satellite is expressed as $\bar{y}$.

Third, the sensing matrix is computed by Eq.(5.66). The inverse of the matrix is computed by LU decomposition. In order to avoid numerical errors, a matrix $\epsilon\ \mathbf{I}_k$ with a small constant ϵ is added to $\mathbf{X}_{\text{ref}}$ before inverting:

$$\hat{\mathbf{\Phi}} = \mathbf{Y}_{\text{ref}}\ (\mathbf{X}_{\text{sub}} + \epsilon\ \mathbf{I}_k)^{-1} \tag{5.72}$$

Note that $\mathbf{I}_k$ refers to an identity matrix of size $k \times k$.

Finally, the target image is estimated by solving Eq.(5.69) based on the estimated sensing matrix $\hat{\mathbf{\Phi}}$ and the light curve $\bar{y}$. The error constraint is 10^{-10}. If the reconstructed image is noisy, the noise is removed by sparse kSVD algorithm with a 2D patch of size 5×5 and a 2D-DCT matrix of size 100×100. The error constraint is $\varepsilon = 28$.

Results

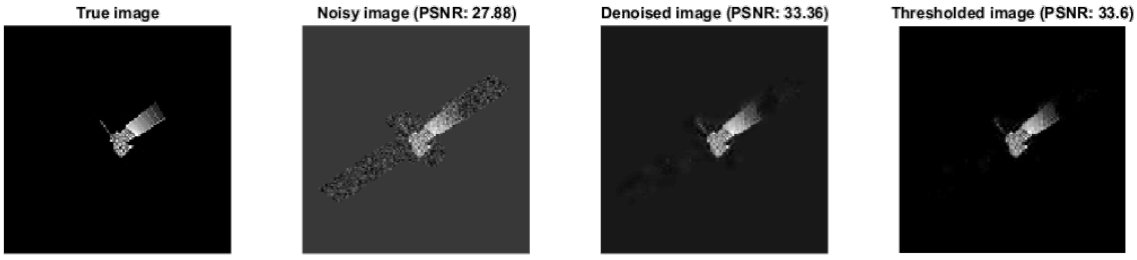


Fig. 5.32.: Comparison between the original image, the reconstructed noisy image, denoised images

In this simulation, a target satellite image is reconstructed based on its light curve and a set of light curves and images of a reference satellite. It is assumed that the reference satellite is always observed from a constant direction and it overlaps with the target satellite in an image. In Figure 5.32, the leftmost one is the original target image and the second one is the result directly obtained by this method. This result contains noise in the shape of the reference satellite . By applying the sparse

KSVD algorithm, the noise is removed and the PSNR value has increased by 5.48 dB as shown in the third image. However, this result still contains some noise. Thus, all the pixels under the threshold $T = 0.4$ are nullified and the rightmost image is obtained. The PSNR value has further increased by 0.24 dB and the result is almost identical with the true image.

6. SUMMARY

6.1 Conclusions

This research discusses a potential application of a compressed sensing technique to characterize an unknown stabilized satellite in a known orbit from its non-resolved light curve via imaging. Characterization is an essential step in space situational awareness, and its objective is to determine the size, shape and attitude of the unknown satellite. A satellite image is one of the most comprehensive ways of satellite characterization and easy to interpret for humans and machines, although a stable attitude has to be given. One of the approaches in the characterization which resembles this research is a radar imaging approach but this approach can only deal with objects that are in low altitude orbits.

Compressed sensing was developed for efficient signal compression and reconstruction. It enables to reconstruct an image from only a fraction of its linear measurement. This measurement is done by an inner product with a random Gaussian matrix, which is referred to as a sensing matrix. One of the applications of compressed sensing, a single-pixel camera, is of great importance in this research. In this camera, the sensing matrix is substituted by a collection of microscopic mirrors. These mirrors reflect the light fluxes from an object either away from or toward a photo-diode. Therefore, only a part of the light fluxes is randomly collected and measured. This measurement can be exploited to reconstruct an image of the object.

The measurement of this single-pixel camera has an analogy with a light curve measurement that is corrupted via atmospheric noise. The light curve is a time history of the brightness of the light reflected by a space object. It is regarded as a summation of all the light fluxes from the object but they are randomly degraded by atmospheric noise. Therefore, the atmospheric noise is regarded as a sensing matrix,

and the light curve is considered as a linear measurement obtained via this unknown sensing matrix. Due to this analogy, a noisy light curve may be exploited to estimate an image of a target satellite. In this thesis, light curve measurements of the target satellite in high altitude orbits have been simulated to test this idea.

There are two main differences between compressed sensing and the light curve. First, the light curve is not exactly identical to a compressed image. The light curve is often simulated by modeling the target satellite as a polygon that consists of small flat facets referred to as a mesh, and by summing up the intensity of light reflected on these meshes. Therefore, in short, the light curve is modeled in a mesh-wise way. On the other hand, the compressed image is expressed in a pixel-wise way. Therefore, it is impossible to associate the light curve and the compressed image without the knowledge of the corresponding relationship with the meshes and the pixels. However, for simplicity, this research models a light curve to fit into the compressed sensing in a pixel-wise fashion, which is referred to as a simple light curve model. In this thesis, this simple light curve is carefully compared with a realistic light curve. The second difference between the reconstruction of the satellite image from the light curve and using the single-pixel camera of classical compressed sensing is the availability of the sensing matrix. In a light curve measurement, it is almost impossible to know the atmospheric noise, which means the sensing matrix is unavailable. Therefore, the image of the target satellite needs to be estimated in the absence of the sensing matrix.

In the first step to approach the satellite characterization problem from light curves using compressed sensing, the sensing matrix is assumed to be known. In this case, it is confirmed that the target satellite image can be reconstructed from both the realistic light curve and the simple adapted light curve.

To ease the assumptions and to make the simulation much more realistic, in the second step, the sensing matrix is assumed to be unknown. First, a dictionary learning approach has been applied. In the first simulation, the KSVD algorithm (k-singular value decomposition algorithm) with a patch-based method has successfully obtained

a clear image of the satellite. However, this method is unpractical since it requires measurements of patch images of the satellite, which is not available in the light curve. Moreover, the limitations of dictionary learning have been shown. Therefore, in the second simulation, a dictionary learning approach has been applied without the patch-based method. However, it has not been possible to reconstruct the satellite image even when restricting the problem further by assuming a fixed outline of the satellite or modifying the KSVD algorithm under an assumption that the sensing matrix has a sparse representation.

As a subsequent step, it has been assumed that the sensing matrix is still unknown but can be inferred from a previous or simultaneous observation of a known satellite. This means a set of light curves and images of the reference satellite is assumed to be known. In the first case under those assumptions, it is assumed that the images of the reference satellite are abundant enough to construct a full-rank matrix of size $128^2 \times 128^2$. In this case, the sensing matrix and the target image can be estimated accurately. In a more realistic case, only fewer images of the reference satellite would be available. As a result, the reference satellite images are sparse and not sufficient for constructing a full-rank matrix. However, utilizing the sparsity property, the target satellite image estimation is achieved. This has been done by using a three-dimensional sparse KSVD algorithm. However, this methodology is found to be very computationally intensive. For the faster computation, it has been shown that under the conditions that the reference satellite is always observed from a constant direction and it overlaps with the target satellite in an image, a computational speedup is achieved while maintaining a similar accuracy in the reconstructed target satellite image.

6.2 Future Work

First of all, further validation is required for the light curve model via a comparison with actual measurements, e.g., from the Purdue Optical Ground Station,

to explicitly show that the simple light curve model in this research is valid. Potentially, sophisticated statistic models have to be taken into account to match the atmospheric movement resulting in the scintillation in the light curve. The effects on the methodology have to be studied carefully and adaptions are likely necessary.

Moreover, a known reference satellite might not always be available. One potential approach is the further investigation of the dictionary learning methodology. In the current research, this approach is successful only with a patch-based method. The main advantage of the patch-based method is to merge the local image of the satellite to obtain the global image. This kind of multi-scale approach is known to be useful in image processing in general. One way to leverage this technique in a light curve would be to split the light curve into different components, using them for reconstruction and merge the results. This would require more investigation about the optical properties of light curves to judge the feasibility. Another possible approach is the use of a guide star, which is well established in astronomical measurements as a reference instead.

REFERENCES

REFERENCES

[1] Space debris by the numbers. [Online]. Available: https://www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers

[2] D. J. Kessler, N. L. Johnson, J. Liou, and M. Matney, "The kessler syndrome: implications to future space operations," *Advances in the Astronautical Sciences*, vol. 137, no. 8, p. 2010, 2010.

[3] D. Mehrholz, L. Leushacke, W. Flury, R. Jehn, H. Klinkrad, and M. Landgraf, "Detecting, tracking and imaging space debris," *ESA Bulletin(0376-4265)*, no. 109, pp. 128–134, 2002.

[4] D. A. Hope, S. M. Jefferies, M. Hart, and J. G. Nagy, "High-resolution speckle imaging through strong atmospheric turbulence," *Optics express*, vol. 24, no. 11, pp. 12 116–12 129, 2016.

[5] M. Kaasalainen and J. Torppa, "Optimization methods for asteroid lightcurve inversion: I. shape determination," *Icarus*, vol. 153, no. 1, pp. 24–36, 2001.

[6] M. Kaasalainen, J. Torppa, and K. Muinonen, "Optimization methods for asteroid lightcurve inversion: Ii. the complete inverse problem," *Icarus*, vol. 153, no. 1, pp. 37–51, 2001.

[7] B. Calef, J. Africano, B. Birge, D. Hall, and P. Kervin, "Photometric signature inversion," in *Unconventional Imaging II*, vol. 6307. International Society for Optics and Photonics, 2006, p. 63070E.

[8] R. Linares, M. K. Jah, J. L. Crassidis, and C. K. Nebelecky, "Space object shape characterization and tracking using light curve and angles data," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 13–25, 2014.

[9] R. Linares and J. L. Crassidis, "Resident space object shape inversion via adaptive hamiltonian markov chain monte carlo," in *AAS/AIAA Space Flight Mechanics Meeting*, 2016, pp. 2016–514.

[10] R. Furfaro, R. Linares, and V. Reddy, "Shape identification of space objects via light curve inversion using deep learning models," in *AMOS Technologies Conference, Maui Economic Development Board, Kihei, Maui, HI*, 2019.

[11] S. Fan, A. Friedman, and C. Frueh, "Satellite shape recovery from light curves with noise," *amos*, p. 23, 2019.

[12] E. Aguilera, M. Nannini, and A. Reigber, "Multisignal compressed sensing for polarimetric sar tomography," *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 5, pp. 871–875, 2012.

[13] ——, "Wavelet-based compressed sensing for sar tomography of forested areas," *IEEE transactions on geoscience and remote sensing*, vol. 51, no. 12, pp. 5283–5295, 2013.

[14] P. Daponte, L. De Vito, F. Picariello, S. Rapuano, and I. Tudosa, "Compressed sensing technologies and challenges for aerospace and defense rf source localization," in *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE, 2018, pp. 634–639.

[15] J. Ma, "Single-pixel remote sensing," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, no. 2, pp. 199–203, 2009.

[16] J. Ma and M. Y. Hussaini, "Extensions of compressed imaging: flying sensor, coded mask, and fast decoding," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 9, pp. 3128–3139, 2011.

[17] J. Ma and F.-X. Le Dimet, "Deblurring from highly incomplete measurements for remote sensing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3, pp. 792–802, 2009.

[18] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[19] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on signal processing*, vol. 58, no. 3, pp. 1553–1564, 2009.

[20] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce, "Discriminative sparse image models for class-specific edge detection and image interpretation," in *European conference on computer vision*. Springer, 2008, pp. 43–56.

[21] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2691–2698.

[22] K. Engan, S. O. Aase, and J. H. Husøy, "Multi-frame compression: Theory and design," *Signal Processing*, vol. 80, no. 10, pp. 2121–2140, 2000.

[23] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," Computer Science Department, Technion, Tech. Rep., 2008.

[24] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE transactions on image processing*, vol. 21, no. 8, pp. 3467–3478, 2012.

[25] H. W. Babcock, "Adaptive optics revisited," *Science*, vol. 249, no. 4966, pp. 253–257, 1990.

[26] C. E. Max, S. S. Olivier, H. W. Friedman, J. An, K. Avicola, B. V. Beeman, H. D. Bissinger, J. M. Brase, G. V. Erbert, D. T. Gavel *et al.*, "Image improvement from a sodium-layer laser guide star adaptive optics system," *Science*, vol. 277, no. 5332, pp. 1649–1652, 1997.

[27] B. D. Warner *et al.*, *A practical guide to lightcurve photometry and analysis*. Springer, 2006, vol. 300.

[28] S. Fan, C. Frueh, and A. Buzzoni, "A light curve simulation of the apollo lunar ascent module," *AIAA/AAS Astrodynamics Specialist Conference*, no. 5504, 2016.

[29] C. Frueh, *Space Traffic Management [lecture notes]*. Purdue University, 2015.

[30] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *Journal of graphics tools*, vol. 2, no. 1, pp. 21–28, 1997.

[31] Space track.org. [Online]. Available: https://www.space-track.org/auth/login

[32] E. Angel, D. Shreiner *et al.*, *Interactive computer graphics: a top-down approach with shader-based OpenGL*. Boston: Addison-Wesley,, 2009.

[33] E. J. Candès *et al.*, "Compressive sampling," in *Proceedings of the international congress of mathematicians*, vol. 3. Madrid, Spain, 2006, pp. 1433–1452.

[34] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 8, pp. 1207–1223, 2006.

[35] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[36] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[37] A. K. Jain, *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall,, 1989.

[38] M. Elad, *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[39] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[40] *A Mathematical Introduction to Compressive Sensing*, 1st ed., ser. Applied and Numerical Harmonic Analysis, 2013.

[41] D. Takhar, J. N. Laska, M. B. Wakin, M. F. Duarte, D. Baron, S. Sarvotham, K. F. Kelly, and R. G. Baraniuk, "A new compressive imaging camera architecture using optical-domain compression," in *Computational Imaging IV*, vol. 6065. International Society for Optics and Photonics, 2006, p. 606509.

[42] M.-J. Sun, M. P. Edgar, G. M. Gibson, B. Sun, N. Radwell, R. Lamb, and M. J. Padgett, "Single-pixel three-dimensional imaging with time-based depth resolution," *Nature communications*, vol. 7, no. 1, pp. 1–6, 2016.

[43] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[44] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[45] P. Hitczenko and S. Kwapień, "On the rademacher series," in *Probability in Banach Spaces, 9*. Springer, 1994, pp. 31–36.

[46] E. W. Weisstein. Hadamard matrix. [Online]. Available: https://mathworld.wolfram.com/HadamardMatrix.html

[47] ——. Wavelet. [Online]. Available: http://mathworld.wolfram.com/Wavelet.html

[48] E. J. Candès and D. L. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise c2 singularities," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 2, pp. 219–266, 2004.

[49] M. N. Do and M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *IEEE Transactions on image processing*, vol. 14, no. 12, pp. 2091–2106, 2005.

[50] R. R. Coifman, Y. Meyer, S. Quake, and M. V. Wickerhauser, "Signal processing and compression with wavelet packets," in *Wavelets and their applications*. Springer, 1994, pp. 363–379.

[51] S. Mallat and G. Peyré, "A review of bandlet methods for geometrical image representation," *Numerical Algorithms*, vol. 44, no. 3, pp. 205–234, 2007.

[52] O. Bryt and M. Elad, "Compression of facial images using the k-svd algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–282, 2008.

[53] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[54] Nasa's tess discovers its first earth-size planet. [Online]. Available: https://www.nasa.gov/feature/goddard/2019/nasa-s-tess-discovers-its-first-earth-size-planet

[55] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE signal processing magazine*, vol. 24, no. 4, pp. 118–121, 2007.