Shape From Shading Problem Using Deep Learning

Soumyajit Sarkar

Finding A Possible Solution for Shape From Shading Problem Using A Deep Learning Architecture

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Soumyajit Sarkar
[Roll No: CS-1804]

under the guidance of

Dr. Bhabatosh Chanda
Professor
Electronics and Communication Sciences Unit

Indian Statistical Institute
Kolkata-700108, India

July 2020

Dedicated to my family and my guide

CERTIFICATE

This is to certify that the dissertation entitled **"Finding A Possible Solution for Shape From Shading Problem Using A Deep Learning Architecture"** submitted by **Soumyajit Sarkar** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance.
The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Bhabatosh Chanda
Professor,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Prof. Bhabatosh Chanda*, Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata, for his guidance, continuous support, encouragement and motivation.

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work.

Finally, I am very much thankful to my parents for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Soumyajit Sarkar
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

Given a single 2D image, our Objective is to pass that image into a Deep Learning Model and get the 3D shape of that image as output. These tasks have received many attentions recently, however, most of this existing approaches rely on 3D supervision, annotation of 2D images with keypoints , and also training with multiple views of each object instance, also known as Photometric stereo method. My frameworks are all based on the GAN architecture, with some custom made data set and also data from the ShapeNet data set, the models can run only on 2D images. Our model takes in a single image, encodes it into a latent code Z and passes it onto a generator, which the subsequently produces the depth image, which is used to construct the point cloud . The use of point cloud in the output representation, makes it possible for us to exploit the depth image of the object from the training image.

Keywords: *GAN, Depth Image, Shading Information, Point Cloud.*

Contents

Chapter 1

Introduction

1.1 Objective and problem statement

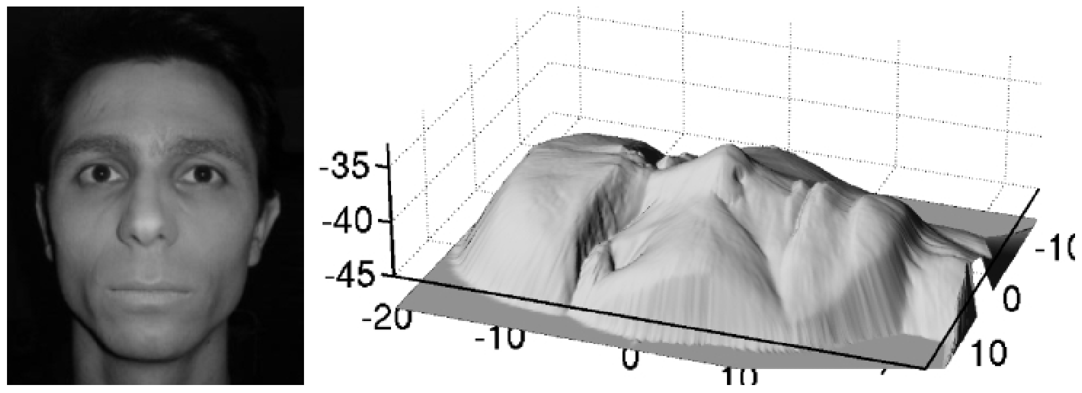


Figure 1.1: 3D reconstruction of a 2D image of a face

Reconstructing 3D objects from 2D images is a long-standing research area in computer vision. While older methods used multiple images of the same object instance, there has been a recent spike in interest of generation 3D model from a single image only, assuming that it shows an object of a class seen during training. A related problem is the generation of new 3D shapes from a given object class a priori, i.e. without conditioning on an image. Again, there have recently been several works that apply deep learning techniques to this task.

Learning-based methods for single-image reconstruction are motivated by the fact that the task is inherently ambiguous: many different shapes project to give the same pixels, for example due to self-occlusion. Hence, we must rely on prior knowledge capturing what shapes are likely to occur. However, most reconstruction methods are trained discriminatively to predict complete shapes from images—they do not represent their prior knowledge about object shapes as an explicit distribution that can generate shapes a priori.

Most learning-based methods for reconstruction and generation rely on strong supervision. This means learning from large collections of manually constructed 3D shapes, it means learning from images paired with aligned 3D meshes, which is very expensive supervision to obtain. While a few methods do not rely on 3D ground-truth, they still require keypoint annotations on the 2D training images , and/or multiple views for each object instance, often with pose annotations.

It is well known that shading provides an important cue for 3D understanding (Horn 1975). It allows determination of surface orientations, if the lighting and material characteristics are known; this has been explored in numerous works on shape-from-shading over the years (Horn 1975; Zhang et al. 1999; Barron and Malik 2015).

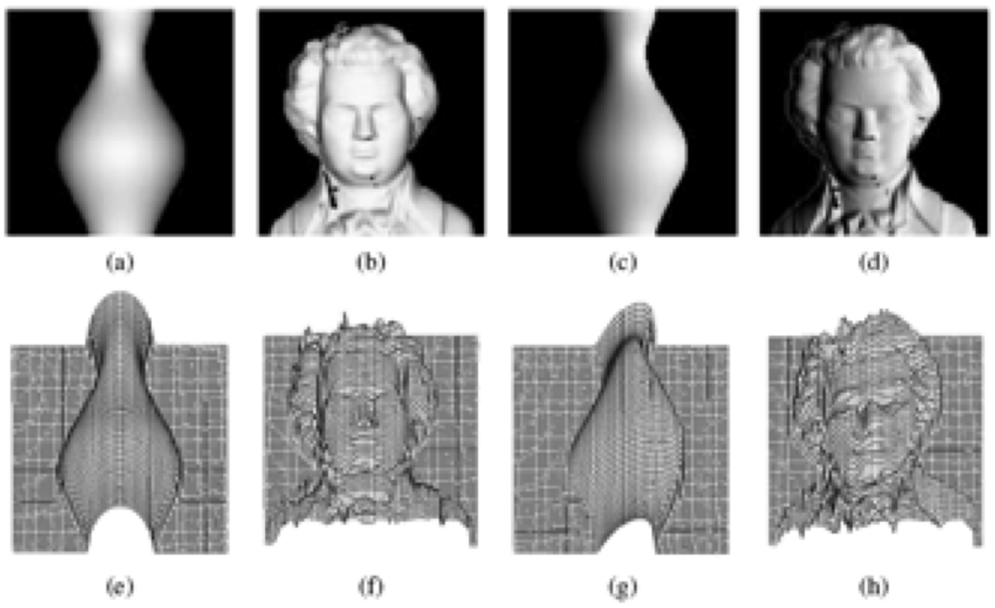


Figure 1.2: Synthetic example of shape from shading application.

Unlike learning-based approaches, these methods can only reconstruct non-occluded parts of an object, and achieving good results requires strong priors. Conversely, existing learning-based generation and reconstruction methods can reason over occluded or visually-ambiguous areas, but do not leverage shading information in their loss.

1.2 Motivation and Application

The main motive for the work is to produce an efficient generative model for the problem of shape from shading for a given Object Class. Since recent work on Deep Learning has seen tremendous rise in research on Generative models and at the same time extremely well performing models, so the purpose was to tap that potential and built a working model for 3D shape generation from a given 2D image. Depth Images are chosen over the traditional albedo and Brightness equation approach, since Depth Images of Objects are now readily available due to advancement in photographic technologies. Taking this route also help bypass the complex mathematics involved in the calculation of the Illumination direction and solving the partial differential equation.

A huge application of this, lies in the determination of Lunar topography and robotics, since this method doesn't require solution to any special equations and complex mathematics, therefore a good enough camera to capture the RGB image is sufficient enough to generate a reasonable estimate of the three - dimensional nature of the surface.

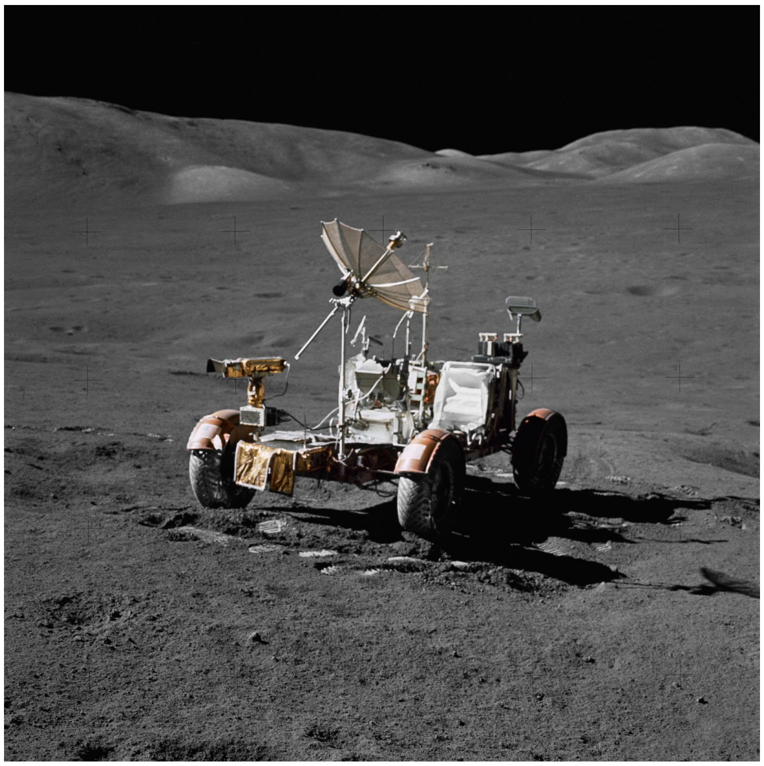
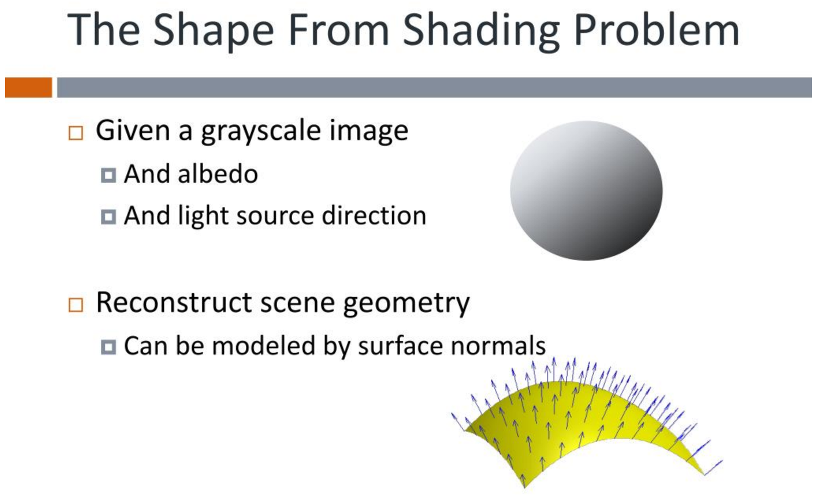


Figure 1.3: Lunar Rover

Judging by our wide use of monocular pictures (photographs or even paintings and woodcuts) of people and other smooth objects, humans are good at interpreting shading information. The shortcomings of our method which are related to the shading information available can be expected to be found in human visual perception too. It will of course be difficult to decide whether the visual system actually determines the shape quantitatively or whether it uses the shading information in a very qualitative way only.

1.3 Organization of this dissertation

The document starts with an introduction to the Shape from Shading problem in Chapter 2, detailing the main statement and the origin of the problem. Then it explains the original Brightness Equation and it's related parts and then proceeds to the explanation of the important terminologies in detail. First it deals with the concept of Surface Normal and then the direction of Illumination, after that a brief description on different type of surfaces are made along with a short note on the Camera Angle. This Chapter ends with a brief discussion on previous related works.



The Shape From Shading Problem

- Given a grayscale image
  - And albedo
  - And light source direction
- Reconstruct scene geometry
  - Can be modeled by surface normals

Figure 1.4: Illustrates shape from Shading Problem [Discussed in detail in Section 2].

In Chapter 3, first of all we go through the basics of Convolution operation and CNN networks, then the basics of GAN is explained and model explored.

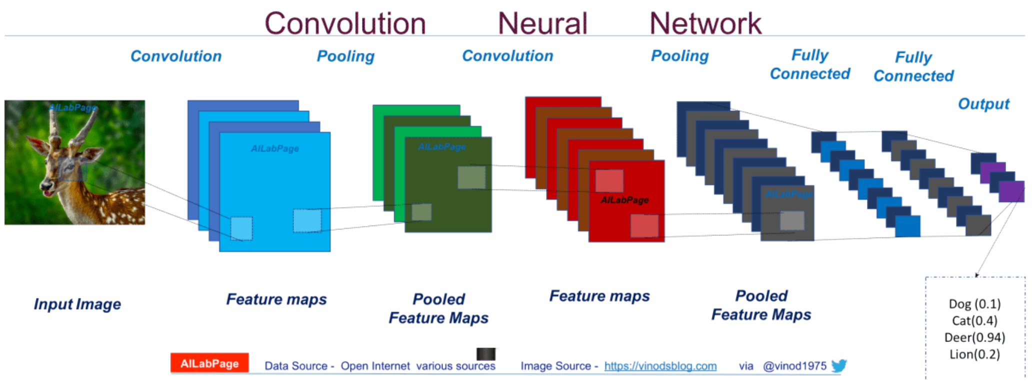


Figure 1.5: A representative figure of CNN [detail discussion is in Section 3].

After that the main model is proposed, the different parts of the model including the Auto-encoder and the Generator and Discriminator is explained visually. Next section contains the Mathematical workings of the model, with an explanation for the Loss Function. The Training section gives details on how the model is to be trained for the best result along with some explanation on the steps taken.

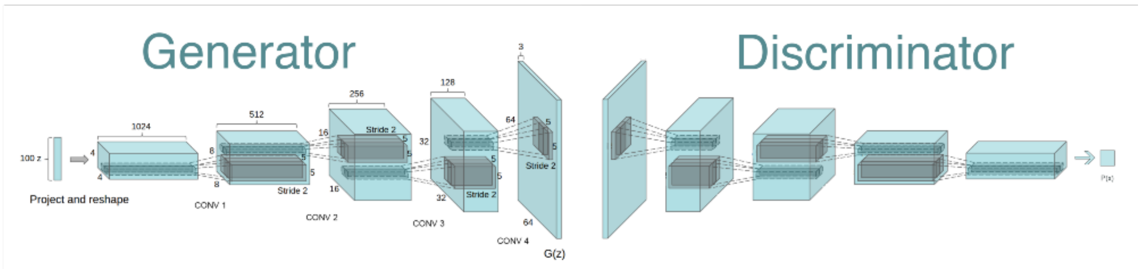


Figure 1.6: A conceptual figure of proposed GAN [detail discussion is in Section 3].

The 4th Chapter, *Experiments*, details the Hardware and software used for the thesis and also gives a detailed explanation of the data sets used for the model. A brief explanation for the Hyper-parameter tuning is also provided in the end, along with results and some Comparisons & Discussions.

Chapter 5 talks about the *future works* possible to tackle the problem and also what other Generative networks can bring on the table along with a brief *Conclusion*.

The document ends with the Bibliography and reference materials.

Chapter 2

The Shape From Shading Problem and Related Works

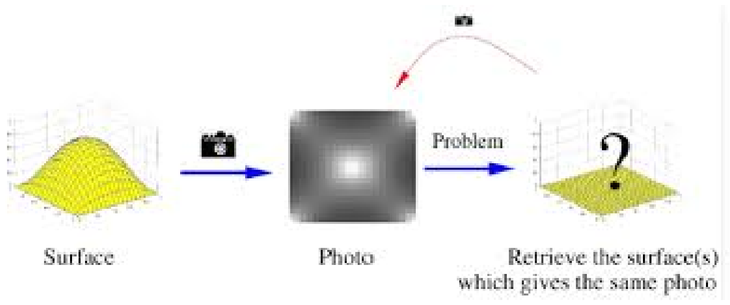


Figure 2.1: The Shape From Shading Problem

Shape From Shading is the process of computing the three-dimensional shape of a surface from one image of that surface. It's different from the other popular three - dimensional reconstruction problems like stereo and photo metric stereo, since here the data is very minimal and as a consequence this problem is intrinsically a difficult one. The shape from shading problem is a ll-posed problem since there is no unique solution.
Horn first formulated the problem in the 70's and tried to solve it by simply modelling it into a non-linear first Order Partial Differential Equation called **brightness equation** and finding a solution for it.

$I(x, y) = R(\vec{n}(x, y))$
Here, (x,y) are the co-ordinates of a pixel; R is the Reflectance map and I is the irradiance.
In the 1980's the computational part of the problem was addressed by directly computing the numerical solutions. In accordance with the Lambertian model of image formation, the grey level at an image pixel depends on the light source direction and

the surface normal. Thus the main aim was to recover illumination source and surface shape at each pixel.
In details, given the image the observed intensity depended on four main factors :

- **Illumination** Illumination defines the position of the incoming light source.
- **Surface reflectivity of the object** This is also known as Albedo. This contains the information about how the object reacts with respect to the incoming light, by giving an idea on the amount of incident light being reflected.
- **Surface Geometry of an Object** This is the objective of the Shape From Shading Problem, given the 2D gray scale image of the object.
- **Camera** Camera captures the object, this can also be meant as the viewer.

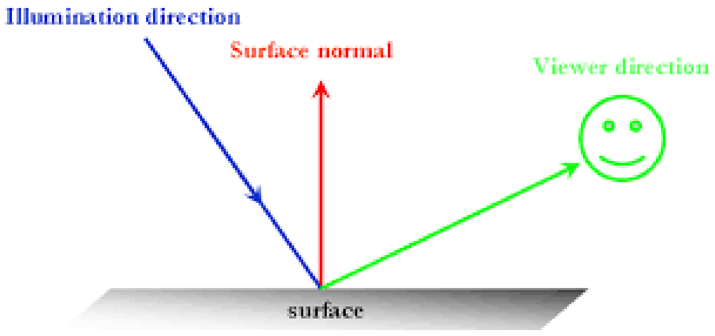
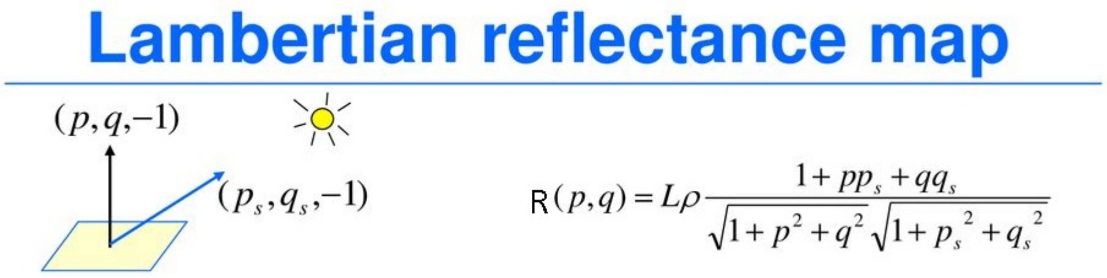


Figure 2.2: Diagram showing the Illumination direction and Surface normal

Given the basic factors on which the image intensity depends, for the Lambertian surface the Reflectance map is defined as the cosine of the angle between the light vector and normal vector to the surface (The normal vector will be explained in details in the next section).



$$R(p,q) = L\rho \frac{1+pp_s+qq_s}{\sqrt{1+p^2+q^2}\sqrt{1+p_s^2+q_s^2}}$$

Figure 2.3: The Lambertian reflectance map

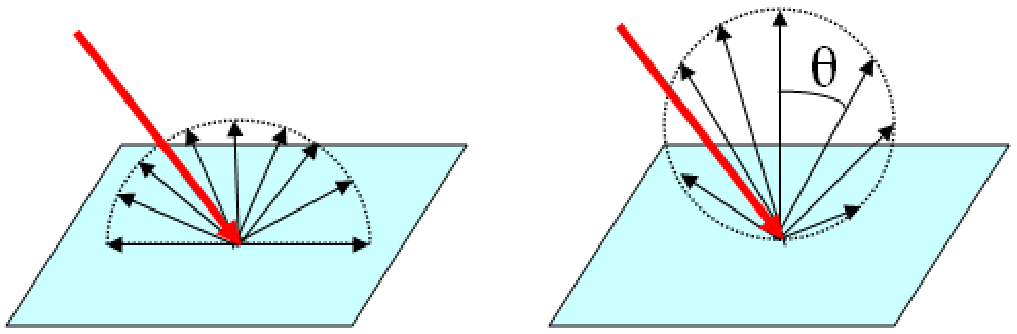


Figure 2.4: Lambertian Surface

The above description of the reflectance map assumed that the surface is ***Lambertian***, now Lambertian surface of reflection is defined as a surface that appears uniformly bright from all directions of view and reflects the entire incident light, that means the surface does not absorb any incident illumination and thus is an ideal surface. The other form of reflectance is Specular reflectance.

The Irradiance is defined as the total energy of radiation falling on a unit area of a surface from all directions above the surface per second. Since energy per unit time is power, we can also say that irradiance is the power incident on a unit area of the surface. Radiance is the outgoing energy ; irradiance is the incoming .

$$I = P/A \tag{2.1}$$

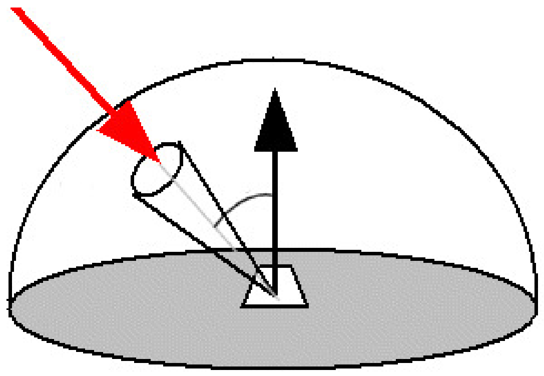


Figure 2.5: Irradiance, Source:MIT

2.1 Terminologies In Detail

Some important Terminologies and concepts for the Model.

2.1.1 Surface Normal

The orientation of the surface can be fully defined by defining the surface normal at each point on the surface. The surface normal is a unit vector perpendicular to the surface.

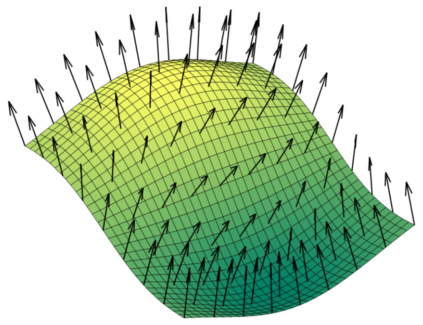


Figure 2.6: Normal Vectors coming out of the surface

In the Spherical co-ordinates, the normal can be defined by its tilt and slant angle, where the tilt (or azimuth angle) is define as the angle between the x-axis and the projection of n on the xy-plane, and the slant (or polar angle) is defined as the angle between the z-axis and the normal. Another representation of surface orientation is by its gradient, using the Tangent Plane.

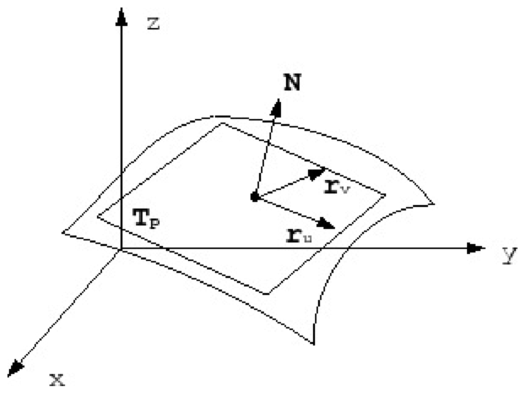


Figure 2.7: Surface normal and Tangent plane

2.1.2 Illumination Direction

The direction from which the object is illuminated. We can represent the illumination direction in terms of its slant and Azimuth angle, in the spherical Co-Ordinate System.

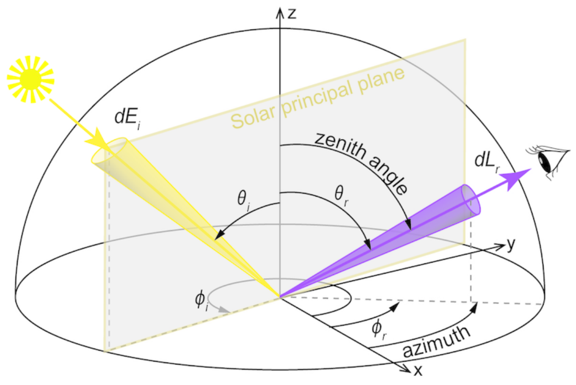
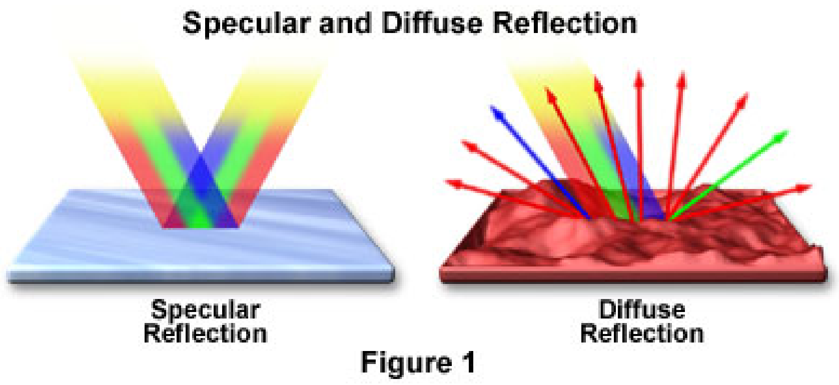


Figure 2.8: Incident and Reflected angles in the Spherical Co-ordinate System

2.1.3 The Type of Surface



Surface reflectivity is the fraction of incident light reflected by the surface. In general, surface reflectivity is a function of the reflected direction and the incident direction and hence it is a directional property. Based on the surface reflectivity, most of the surface can be divided into two parts :

- Diffuse: For diffuse surfaces, such as matte white paint, reflectivity is uniform, light is reflected in all angles equally or near-equally, such surfaces are said to be Lambertian. Most real objects have some mixture of diffuse and specular reflective properties.

- Specular: For specular surfaces, such as glass or polished metal, reflectivity will be nearly zero at all angles except at the appropriate reflected angle where the surface shines

For a Lambertian surface, the intensity only depends on the angle between the illumination direction and the surface normal (α), hence under Lambertian assumption, the reflectance at point (x,y,z) is a function of the angle between the illumination direction α and the surface reflectivity (albedo)ρ

$$R = f(\alpha, \rho) \tag{2.2}$$

2.1.4 Camera (The Viewer)

A camera is a projective system, it projects 3D point to 2D image pixels. To model the camera process, we use projections models such as perspective and orthographic projection. Under the assumption of orthographic projection, a point (X,Y,Z) can be projected to a 2D point (x,y) such that x = mX and y = mY, where m is a magnification factor of our choice. While using perspective projection, the 2D point will be defined as x = X/Z and y = Y/Z

2.2 Related Works

There has been a surge of interest in single-image 3D reconstruction, this has been enabled both by the growing maturity of deep learning techniques, and by the availability of large data sets of 3D shapes. Among such methods, we differentiate between those requiring full 3D supervision (i.e. 3D shapes paired with images), and those that need only weaker 2D supervision (pose annotations).

- **3D-Supervised Methods:** Choy et al. (A unified approach for single and multi-view 3D object reconstruction, 2016) apply a CNN to the input image, then pass the resulting features to a 3D deconvolutional network, that maps them to to occupancies of a 32^3 voxel grid. Girdhar et al.and Wu et al. (2016) both proceed similarly, but pre-train a model to auto-encode or generate 3D shapes respectively, and regress images to latent features (z). Instead of directly producing voxels, Soltani et al. (2017), Shin et al. (2018) and Richter and Roth (2018) output multiple depth-maps and/or silhouettes, from known (fixed) viewpoint. Fan et al. (2017) and Mandikal et al. (2018) generate point clouds as the output, with networks and losses specialised to their order invariant structure. Tulsiani et al. (2017a) and Niu et al. (2018) both learn to map images to sets of cuboidal primitives, of fixed and variable cardinality respectively. As in the previous works, they require large numbers of 3D shapes and.

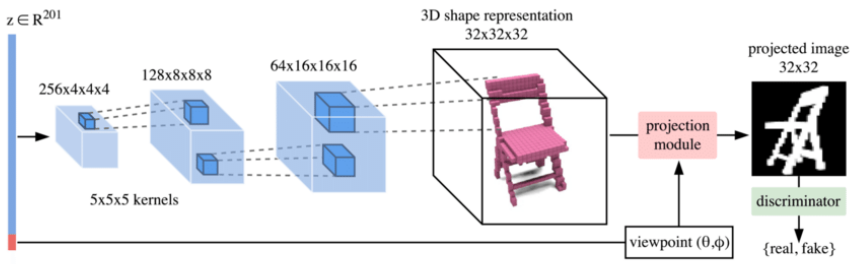


Figure 2.9: The PrGAN architecture for generating 3D voxel representation of 2D images

- **2D-Supervised Methods :** This 2D Supervsed models generally work by passing input images through a CNN, which predicts a 3D representation, which is then rendered to form a reconstructed 2D silhouette; the loss is defined to minimise the difference between the reconstructed and original silhouettes. All these methods require stronger supervision, they must be trained with ground-truth pose or keypoint annotations, and/or multiple views of each instance presented together during training.

  In a 2016 paper by Rezende et al, briefly discuss single-image reconstruction using a conditional generative model over meshes. This models radial offsets to vertices of a spherical base mesh, conditioning on an input image. The model is trained in a variational framework to maximise the reconstructed pixel likelihood. It is demonstrated only on simple shapes such as cubes and cylinders.

  Yan et al. (2016) present a method that takes single image as input, and yields a voxel reconstruction. This is trained to predict voxels that re project correctly to the input pixels, assuming the object poses for the training images are known. The voxels are projected by computing a max operation along rays cast from each pixel into the voxel grid, at poses matching the input images. The training objective is then to maximise the intersection-over-union (IOU) between these projected silhouettes and the silhouettes of the original images. Kato et al. (2018) present a very similar method, but using meshes instead of voxels as the output representation.

Chapter 3

Deep Learning Model Proposed

Our Goal is to build a Probabilistic Generative model that can generate a 3D Shape from a given 2D image for some certain object class. This whole process require 3 key ingredients :

- **The original RGB image**
- **Corresponding Depth Image**
- **3D Data Representation**

The Goal of our model is to generate the Depth Image given the original RGB image as input. The Depth image is then used for the 3D data Representation, now there are multiple ways Image Data can be represented in 3D, like Voxel grids, Point Clouds and Polygonal Mesh, here in the model we will use Point Clouds to represent the end result of our generated Depth Image.

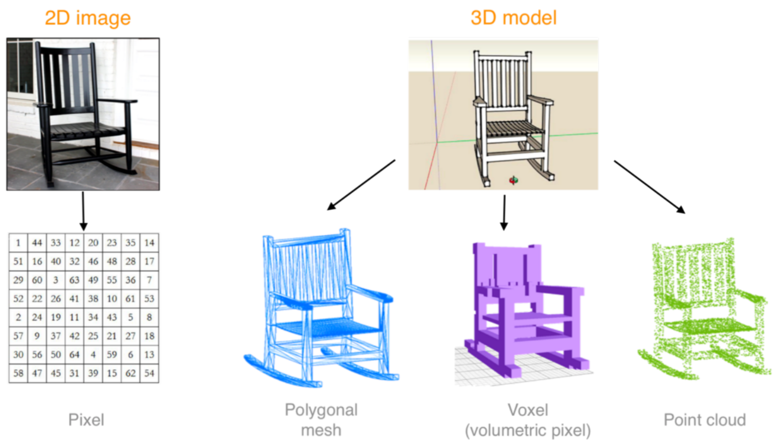


Figure 3.1: Ways to represent a 2D image in three-dimension.

Point clouds are a collection of points in 3D coordinate (x, y, z), together these points form a cloud that resemble the shape of object in 3 dimension. The larger the collection of points, the more details it gets. The same set of points in different order still represents the same 3D object.
This compact representation focuses on the detail surfaces of the 3D objects but a lack of direct support of the CNN is a disadvantage.

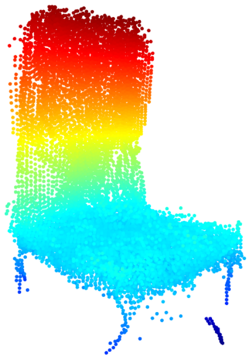


Figure 3.2: Point cloud representation of a chair

3.1 CNN and GAN

3.1.1 CNN

CNN stands for Convolutional Neural Network, it's a class of Deep Neural Networks used for analyzing and processing images. It's mainly used for images recognition, images classifications. Objects detections, recognition faces etc.

The basic 2D CNN operation comprises of extracting important details from the input image, it follows the function :

$$(f * g)(t) \overset{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{3.1}$$

In the Convolution Operation a filter matrix slides through the Image pixel matrix, taking in the product and sum value along the way and forming an image with reduced pixel size and only containing the important bit of information.

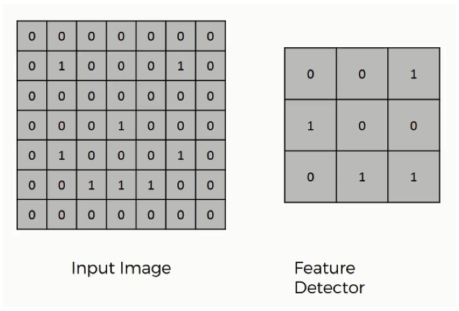


Figure 3.3: The Image and the feature map

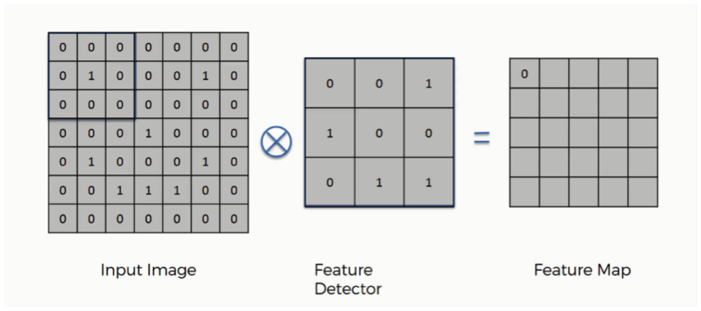


Figure 3.4: The Convolution Operation, and feature map generation

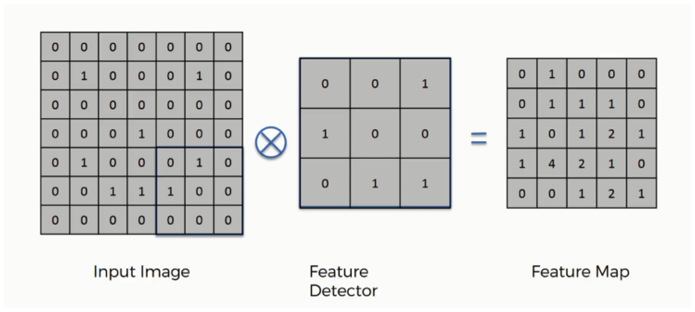


Figure 3.5: The Final Result of the Convolution Operation

3.1.2 GAN

GAN: Generative **A**dversarial **N**etwork

GAN is a Deep Learning Model that consists of 2 Networks, a Generator and a Discriminator.

The Generator learns to generate plausible data that becomes task of Discriminator to detect and The Discriminator learns to distinguish the generator's fake data from real data. The discriminator penalizes the generator for producing implausible results.

On the start of training, the input to the generator is generally a 100 dimensional vector consisting of random noise taken from a normal distributed system. The generator produces fake data and obviously the discriminator learns to tell that it's fake but as training progresses, the generator gets closer to producing output that can fool the discriminator.

Finally, if generator training goes well, the discriminator gets worse at telling the difference between real and fake. It starts to classify fake data as real, and its accuracy decreases, then we can say that Nash Equllibrium is reached and the training process stops.

The Whole pipeline roughly looks like :

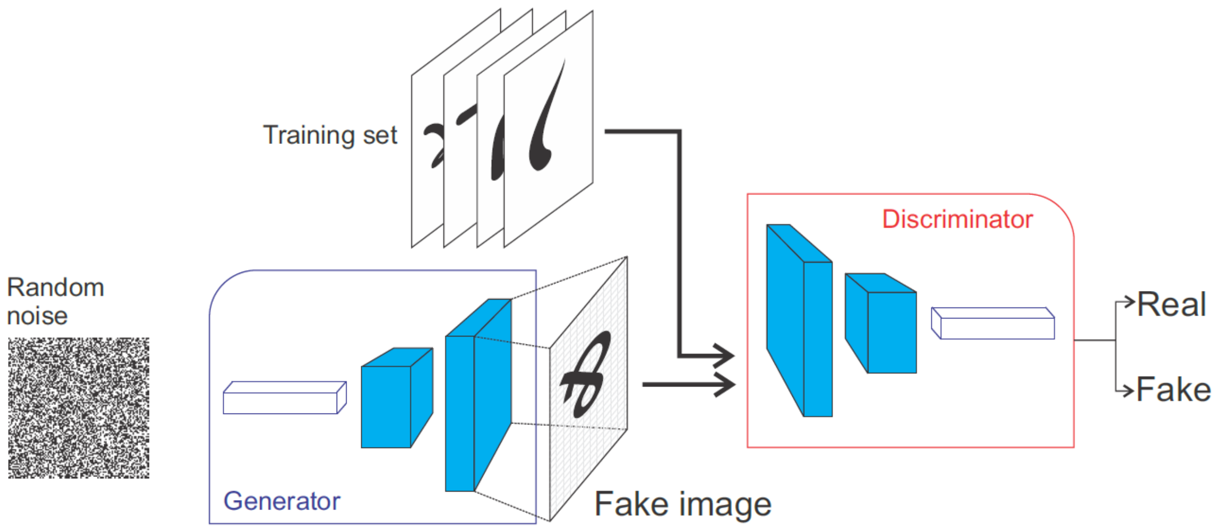


Figure 3.6: GAN Model

3.1.3 Auto-encoder

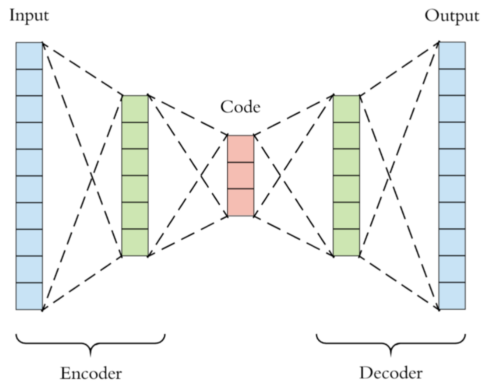


Figure 3.7: Auto-encoder

Auto-encoders are feed forward neural networks with same input and output. It takes an image as an input and compresses it into a lower dimensional code, so that it can be compressed and also reconstructed back from it. The Auto-encoder in our model takes the gray-scale image of the input RGB image and transforms into a 100 dimensional vector, Z i.e it compresses it.

The Auto-encoder consists of 3 components : **Encoder**, **Code(Z)**, **Decoder**. Since the input and output are the same images, this is not really supervised or unsupervised learning, it's typically called self-supervised learning. Mean squared Error is the loss function used. Mathematically,

$$\overbrace{\phi}^{\text{Encoder}} : X \longrightarrow Z \tag{3.2}$$

$$\overbrace{\psi}^{\text{Decoder}} : Z \longrightarrow X \tag{3.3}$$

$$\phi, \psi = \underset{\phi, \psi}{argmin} \, ||X - (\psi \circ \phi)X||^2 \tag{3.4}$$

Z is the latent space, presented at the bottleneck. The decoded function maps the latent space Z to the output. Since the output is same as the input, therefore it's just recreation of the original image after some generalized non-linear compression. Latent Z can be presented by,

$$Z = \sigma(Wx + b) \tag{3.5}$$

The decoded output,

$$x' = \sigma'(W'Z + b') \tag{3.6}$$

Therefore the loss function in terms of the network functions,

$$L(x, x') = ||x - x'||^2 = ||x - \sigma'(W'(\sigma(Wx + b) + b')||^2 \tag{3.7}$$

3.2 Proposed Model Architecture

3.2.1 Model Design

The model is based on the DCGAN Architecture. The input to the model is a vector Z of size 100, this vector contains the encoded information of the original RGB image, which we convert to a gray scale image and encode it into a vector of size 100.

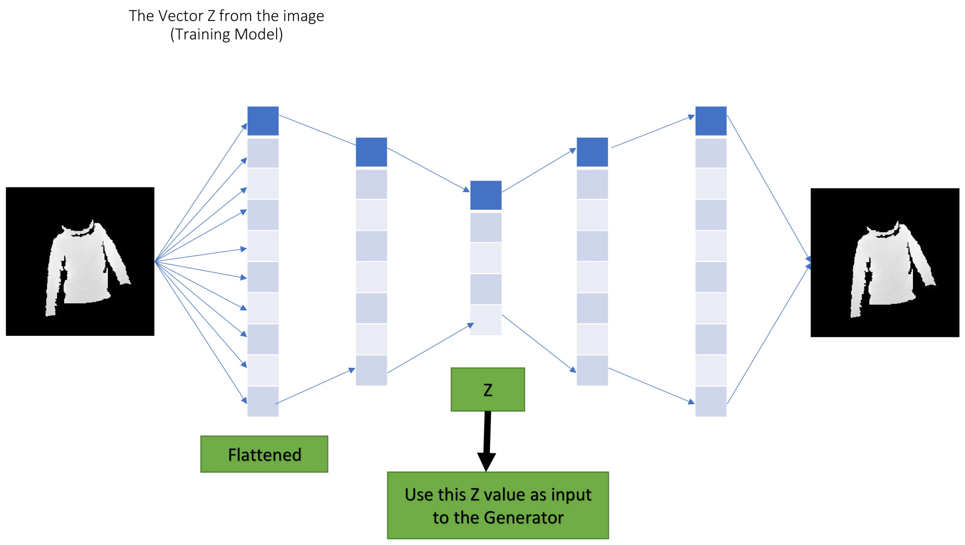

The Vector Z from the image
(Training Model)
Z
Flattened
Use this Z value as input to the Generator

Figure 3.8: The image encoder model

The input to the Generator is the vector Z, and the model uses standard DCGAN architecture of the Generator.

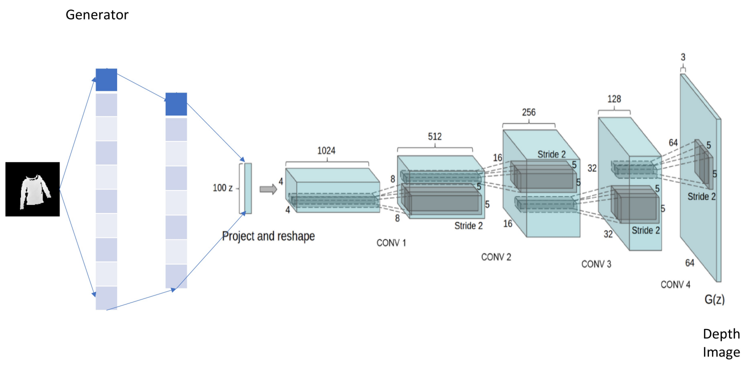


Figure 3.9: The Generator

The Generated Depth Image is then fed to Discriminator for Probabilistic Analysis of whether it's a fake or not and the resulted loss is then used to update the weights of both the Generator and the Discriminator.

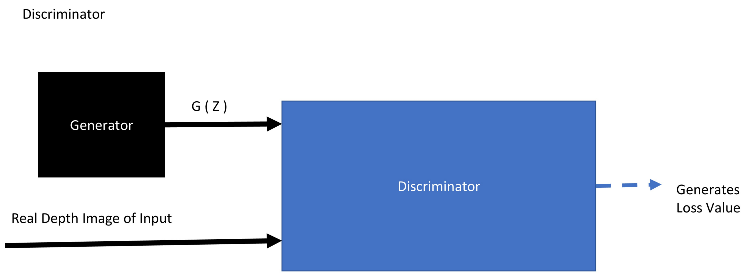


Figure 3.10: Discriminator

Finally the Depth Image is converted to a Point Cloud 3D representation using Open3D library. The total model pipeline

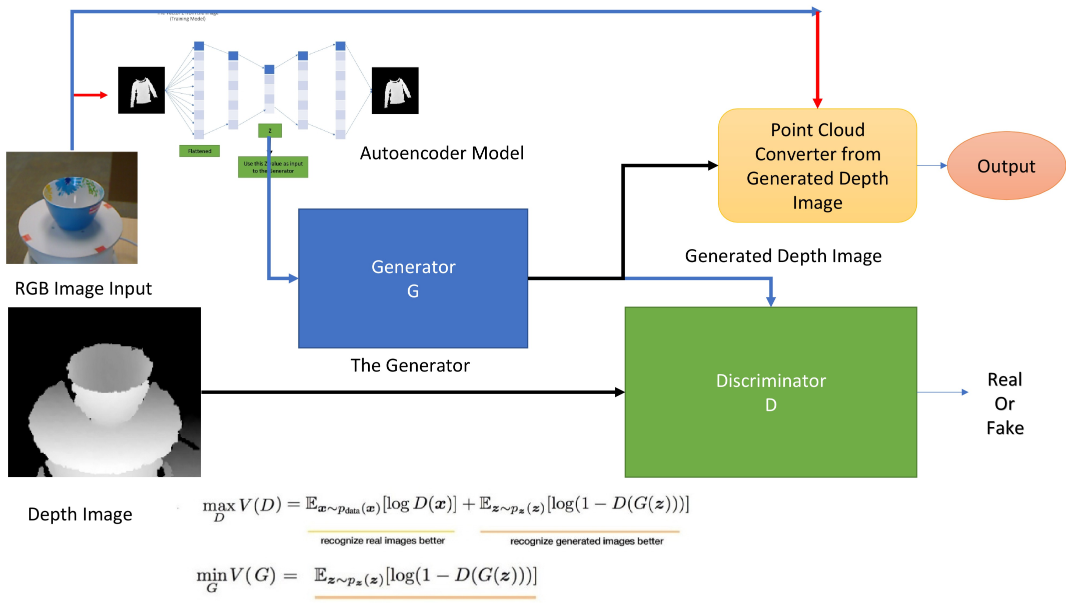

Autoencoder Model
Flattened
Use this Z value as input to the Generator
Point Cloud Converter from Generated Depth Image
Output
RGB Image Input
Generator G
The Generator
Generated Depth Image
Discriminator D
Real Or Fake
Depth Image
$\max_D V(D) = \underbrace{\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)]}_{\text{recognize real images better}} + \underbrace{\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]}_{\text{recognize generated images better}}$
$\min_G V(G) = \underbrace{\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]}$

Figure 3.11: Model Pipeline

3.2.2 Model Mathematical Explanation

The model contains two parts :

- **Generative Network:** It's a model that given inputs can understand the inputs to generate similar inputs and it's labels from the targets.
  The Model learns the joint probability distribution.

  $$P(x, y) = p(x|y).p(y) \tag{3.8}$$

  where $P(x|y)$ = Probability of x given y should be maximum or y happened. The model has to learn the $p(x|y)$ and it tries to understand and learn, how the training data is generated/distributed.

- **Discriminative network:** It's a model that given an input, classifies the inputs to the corresponding targets as correct as possible.
  It learns the Conditional Probability Distribution.

  · P($y|x$) = Probability of y given x should be maximum or x happened.

  The model learns to predict the labels from the data, in other words it learns the decision boundary between classes. It's not interested in knowing about how the training data is generated/distributed.

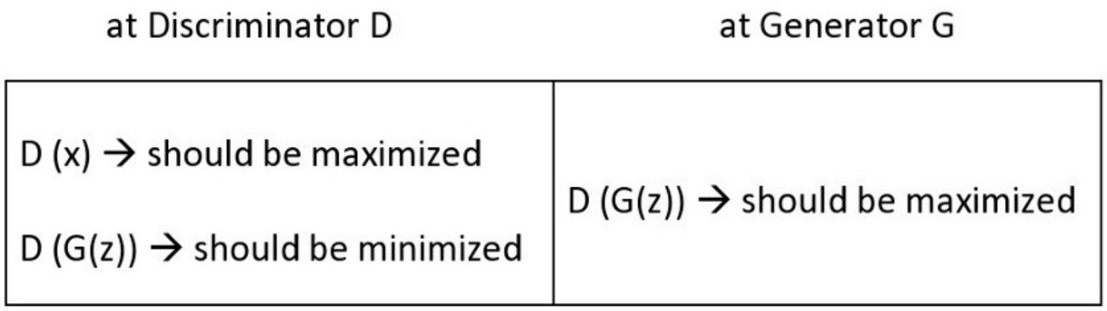
Our Model tries to learn the input image distribution and generate the corresponding depth images as realistic as possible. The discriminator identifies whether the image generated is a good fake or not and accordingly the process continues until an Equilibrium is reached.

As the discriminator is a binary classifier so when we feed the real data or the original depth images , the model should produce high probability for the real data and low probability for fake depth images(generator's output).

- **Z** : Input Vector
- **G(Z)** : Generator's output, the generated depth images i.e $\mathrm{x}_{\mathrm{fake}}$
- **x**: Training Sample, the original depth images $\mathrm{x}_{\mathrm{real}}$
- **D(x)**: Discriminator output for $\mathrm{x}_{\mathrm{real}}$

- **D(G(Z))**: Discriminator's output for $\mathrm{x}_{\mathrm{fake}}$

Now, **D(x)** actually represents the value of $P(y|x_{\mathrm{real}})$ and the value of **D(G(Z))** represents the value of $P(y|x_{\mathrm{fake}})$, therefore they have scores between 0 and 1. Therefore our main objective is to :



at Discriminator D	at Generator G
D (x) → should be maximized D (G(z)) → should be minimized	D (G(z)) → should be maximized

Therefore, the model (discriminator) must maximizes the real data while minimizing the fake data and also the model(generator) must maximizes the fake data, until Equilibrium is reached. The Loss function values for the Generator and Discriminators are as follows.

Loss at Discriminator **D**

$$\mathbf{Dloss_{real}} : log(D(x)) \tag{3.9}$$

$$\mathbf{Dloss_{fake}} : log(1 - D(G(Z))) \tag{3.10}$$

Therefore Using Equation 3.3 and 3.4, we have

$$\textbf{Net Dloss} : Dloss_{\mathrm{real}} + Dloss_{\mathrm{fake}} i.e[log(D(x)) + log(1 - D(G(Z)))] \tag{3.11}$$

Therefore for a batch of **m** data sets, the total loss for the Discriminator is:

$$\frac{1}{m}\sum_{i=1} log(D(x^i)) + log(1 - D(G(Z^i))) \tag{3.12}$$

Loss at Generator **G**

$$Gloss : log(1 - D(G(Z))) \tag{3.13}$$

Therefore for a batch of **m** data sets, using Equation 5 the total loss for the Generator is:

$$\frac{1}{m}\sum_{i=1} log(1 - D(G(Z^i))) \tag{3.14}$$

The discriminator network runs twice (one for real, one for fake) before it calculates the final loss while generator runs only once. Once the two losses are calculated, we calculate the gradients w.r.t to their parameters and back propagate through their networks independently, using a suitable optimizer.
The model tries to minimize and maximize the value of the function V at the same time, corresponding to the Generator and Discriminator. Using Equation 3.3,3.4 and 3.7 we have :

$$\underset{\mathbf{G}}{min}\,\underset{\mathbf{D}}{max}\, V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{x\sim p_z(z)}[log(1 - D(G(z)))] \tag{3.15}$$

more briefly, we have for Discriminator and Generator respectively :

$$\overbrace{\underset{\mathbf{D}}{max}\, V(D)}^{\text{Discriminator}} = \underbrace{E_{x\sim p_{data}(x)}[logD(x)]}_{\text{recognize real images better}} + \underbrace{E_{x\sim p_z(z)}[log(1 - D(G(z)))]}_{\text{recognize generated images better}} \tag{3.16}$$

$$\overbrace{\underset{\mathbf{G}}{min}\, V(G)}^{\text{Generator}} = E_{x\sim p_z(z)}[log(1 - D(G(z)))] \tag{3.17}$$

3.3 Training

Now that all the frameworks are defined, the model training process is explained .Here, the Algorithm from Ian Goodfellow's paper [2014] is followed closely. Training is split up into two main parts. Part A updates the Discriminator and Part B updates the Generator.

Part A: Training the Discriminator

The goal of training the discriminator is to maximize the probability of correctly classifying a given input as real or fake, the discriminator is to be updated by ascending its stochastic gradient. We want to maximize [**log(D(x))+log(1-D(G(Z)))**]
firstly, a batch of real samples is constructed from the training set, forward pass through **D**, the loss **(log(D(x)))** is calculated, then the gradients are calculated in a

backward pass.
Secondly, a batch of fake samples is constructed with the current generator, the batch is then passed through **D**, the loss [**log(1-D(G(Z)))**] calculated, and gradients accumulated with a backward pass. Now, with the gradients accumulated from both the all-real and all-fake batches, Discriminator's optimizer is called into the scene.

Part B: Training the Generator

The Generator is trained keeping in mind the objective of minimizing **log(1-D(G(Z)))**, in an effort to generate better fakes. **This is shown by Goodfellow to not provide sufficient gradients, especially early in the learning process**. As a fix, maximize **log(D(G(z)))** is done. It's actually accomplished by: classifying the Generator output from Part A with the Discriminator, computing G's loss using real labels as GT, computing G's gradients in a backward pass, and finally updating G's parameters with an optimizer step. It may seem counter-intuitive to use the real labels as GT labels for the loss function, but this allows the use of the log(x) part of the BCELoss(Used in the model) (rather than the $log(1-x)$ part).

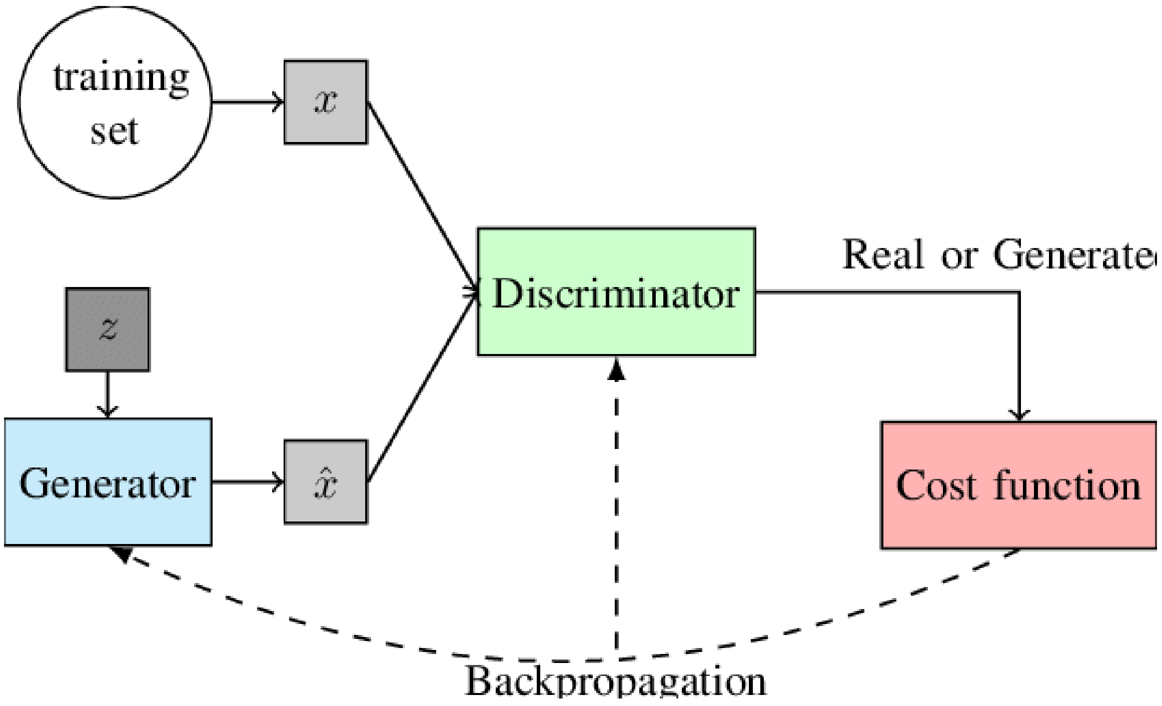


Figure 3.12: GAN training overview, Source : DeepAI

Chapter 4

Experiments

4.1 Hardware and Software used

4.1.1 Hardware

The Hardware Used for Thesis Work : Server Machine

Figure 4.1: Server Machine

Specifications :

- Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz
- Architecture : x86
- Linux Ubuntu 16.04.4 LTS
- NVIDIA GTX TITAN Xp (CUDA Version 10.1)
- 128818 MB Memory

4.1.2 Software

Language used to design the model architecture : **Python**

Deep Learning Framework used to build the model using Python : **PyTorch**

Figure 4.2: Official PyTorch Logo.

PyTorch is a open source Machine Learning and Deep Learning Library, build on the popular torch library. It is mainly used for applications in Computer Vision and Natural Language Processing and it's primarily developed by Facebook AI Research Lab.

4.2 Dataset

Objects from 2 different Data sets are used in total. The common thing in both the data sets is the type of content, i.e all of them had the original Image and the corresponding Depth Image.

- **RGB-D Object Data set** The RGB-D Object Dataset is a large dataset of 300 common household objects. The objects are organized into 51 categories arranged using WordNet hypernym-hyponym relationships.This dataset was recorded using a Kinect style 3D camera that records synchronized and aligned 640x480 RGB and depth images at 30 Hz.

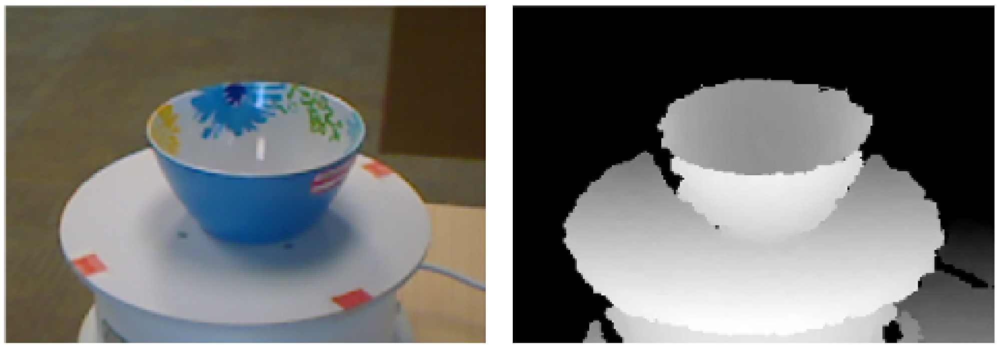


Figure 4.3: RGB-D Dataset

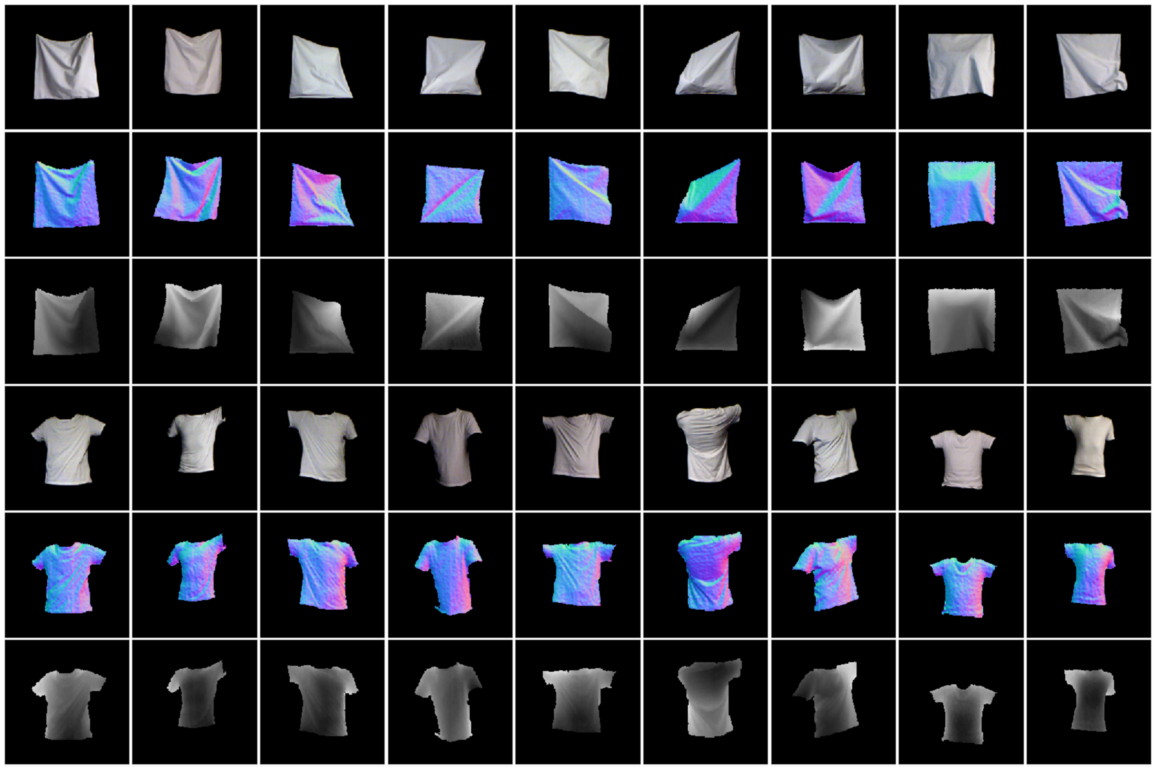


Figure 4.4: Textureless Deformable Surface Dataset

- **Textureless Deformable Surface** The dataset features deformable objects with uniform albedo captured under varying lighting conditions. The dataset was captured using Microsoft Kinect Xbox 360 which provides both the RGB images and corresponding depth maps with resolution 640 x 480 px.

4.3 Parameter Tuning

- **Batch Size :** Batch Size can be tested with batches of 8, 16, 32, 54, or 128, although it's better to have lower batch size since using a bigger batch size might hurt the performance because during the initial training the discriminator might get a lot of examples to train on and it might overpower the generator, which would have a negative effect on training.

- **The Optimization Algorithm :** Different Optimizers to experiment with like Adam, SGD, Adadelta, RMSProp and other optimizers available.

- **Number of Epochs:** Starting with 100 epochs, it can be gradually raised to 1000 to 5000.

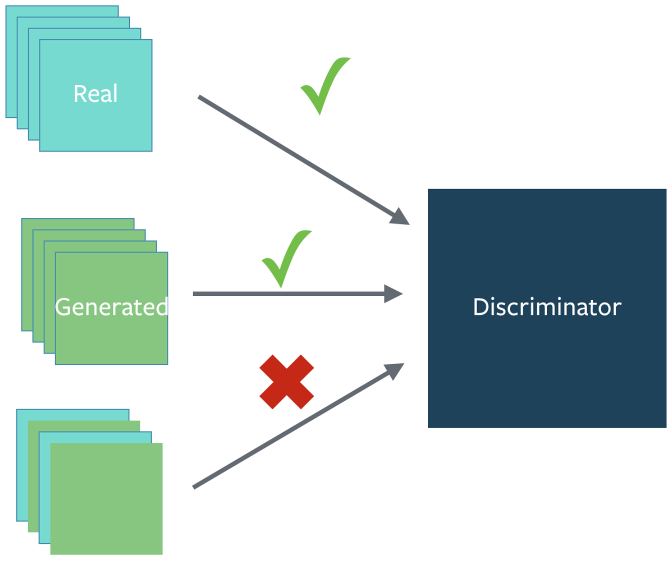


Figure 4.5: No batch mixing, source:ganhacks

- **Different mini-batches :** Different mini batches are constructed for real and fake images i.e each mini-batch contains only all real images or all generated images. No batch mixing.

- **Learning rate :** The most important hyper-parameter, different learning rates are used starting with smaller values like 0.1,0.01 etc. This is done so as to make sure that as we approach the *Minima*, we don't overshoot or start oscillating around the point.

- **Activation functions in different layers of the generator and the discriminator network:** Different Activation Functions can be experimented with, like sigmoid, tanh, ReLU, LeakyReLU, ELU, SeLU, and other activation functions.

- **Loss Functions :** BCELoss is used here but other options are Wasserstein loss function, Relativistic Average Least Square loss function both of which gives significant good results.

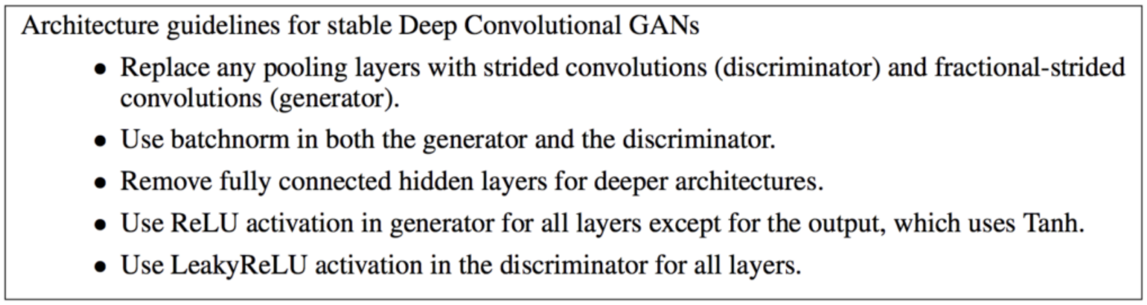
  - *Modified Loss Function :* In the GAN paper, the loss function used was $minlog(1-D)$, but in training, $maxlog(D)$ is used, since the earlier causes the vanishing gradient problem.

- **Number of Layers in the Network :** Value of this parameter is set experimentally by trial-and-error method to achieve best training result.

- **Strided Convolutions :** Instead of pooling layers, Strided Convolutions are used. This is because, as stated in *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, 2015*,

  ”deterministic spatial pooling functions (such as max pooling) with strided convolutions, allowing the network to learn its own spatial downsampling. We use this approach in our generator, allowing it to learn its own spatial upsampling, and discriminator.”

- **Architectural Guidelines:** From the 2015 paper, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, certain architectural recommendations are made for better training of a DCGAN network.


Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Figure 4.6: DCGAN Architectural Recommendations

4.4 Results(To be Updated)

To be Updated

4.5 Comparison and Discussion(To be Updated)

To be Updated

Chapter 5

Conclusion and Future work

5.1 Conclusion

For the dissertation thesis a framework is presented, which can generate and construct a three-dimensional shape from a given 2D RGB image by generating the Depth information of different objects in the image in the form of a Depth map. This approach is much more flexible than the older classical algorithms, as it utilises the excellent processing powers of the modern neural nets and also requires very small amount of Input data, unlike the older algorithms. Since it works with Depth Images and point clouds, therefore information about the illumination direction and albedo is also not required, which we would have required if we had used 3D meshes instead of point clouds.

The Limitation on the model is the confinement on certain object class and not a generalized version. Since GAN can learn one object class at a time and then after multiple such learnings, generates a random image, the model can learn shading information of one object class at a time, and needs further training again for other object class.Weights from one class may not be suitable for generating depth images of another class.

5.2 Future Work

In Future work, more data needs to be collected to make a truly robust network for different Structural environment. Since the model can only generate Depth Images for one class at a time, more work needs to be done on the Generator so as to construct a model that can train on multiple different classes and can successfully differentiate between them and learn their distributions separately. This will be tried with some external informational input to the Generator and then trial and error. Further, tips can be taken from earlier algorithms also.

Bibliography

1. Horn, B. (1975). Obtaining shape from shading information. In P. H. Winston (Ed.), The psychology of computer vision.

2. Learning Single-Image 3D Reconstruction by Generative Modelling of Shape, Pose and Shading by Paul Henderson and Vittorio Ferrari International Journal of Computer Vision. [2020]

3. IsMo-GAN: Adversarial Learning for Monocular Non-Rigid 3D Reconstruction by Soshi Shimada, Vladislav Golyanik, Christian Theobalt, Didier Stricker

4. 3D Shape Induction from 2D Views of Multiple Objects by Matheus Gadelha, Subhransu Maji and Rui Wang, University of Massachusetts, Amherst

5. Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S. (2016). 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In Proceedings of the European conference on computer vision.

6. Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.), Advances in neural information processing systems

7. Soltani, A. A., Huang, H., Wu, J., Kulkarni, T. D., and Tenenbaum, J. B. (2017). Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In Proceedings of the IEEE conference on computer vision and pattern recognition.

8. Shin, D., Fowlkes, C. C., and Hoiem, D. (2018). Pixels, voxels, and views: A study of shape representations for single view 3D object shape prediction. In Proceedings of the IEEE conference on computer vision and pattern recognition.

9. Fan, H., Su, H., and Guibas, L. (2017). A point set generation network for 3D object reconstruction from a single image. In Proceedings of the IEEE conference on computer vision and pattern recognition.

10. Mandikal, P., Murthy, N., Agarwal, M., and Babu, R. V. (2018). 3D-LMNet: Latent embedding matching for accurate and diverse 3D point cloud reconstruction from a single image. In Proceedings of the British machine vision conference.

11. Rezende, D. J., Ali Eslami, S. M., Mohamed, S., Battaglia, P., Jaderberg, M., and Heess, N. (2016). Unsupervised learning of 3D structure from images. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.)

12. Kato, H., Ushiku, Y., and Harada, T. (2018). Neural 3D mesh renderer. In Proceedings of the IEEE conference on computer vision and pattern recognition.