**Ph.D. Thesis**

Université de Limoges


## University of Limoges

### ED 653 : SCIENCES ET INGENIERIE

**XLIM Research Institute – UMR CNRS 7252**


A thesis submitted to the University of Limoges
in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

Image Synthesis and Analysis


Presented and defensed by

**Shahrzad Pourmand**


On February 3, 2023

# Simultaneous Acquisition of Geometry and Reflectance

Thesis supervisors:

Mr. Stéphane Mérillou and Mr. Nicolas Mérillou

JURY:

President of jury
Mr. Jean-Jacques Bourdin — Professor, University of Paris 8

Reporters
Mr. Jean-Pierre Jessel — Professor, University of Toulouse-III Paul-Sabatier
Mr. Guillaume Gilet — Professor, University of Sherbrooke

Examiners
Mr. Maxime Maria — Professor, University of Limoges

Dedicated to my family

*I am so clever that sometimes I don't understand a single word of what I am saying.*
**Oscar Wilde**

# Acknowledgements

First of all, I would like to express my deepest gratitude to my Ph.D. supervisors, Mr. Stephane Mérillou and Mr. Nicolas Mérillou, for their guidance and support. I am very thankful for their help and mentoring throughout my Ph.D. studies. In addition, I would like to thank the jury members for taking the time to read and evaluate my work.

I would also like to thank the members of the International Welcome Office of the University of Limoges (Bureau d'Accueil International) for their assistance related to my stay.

Moreover, I would like to thank my uncle and aunt, Vahid and Mitra Meghdadi, as well as my cousins, for their continuous support and encouragement during my stay in Limoges. I would also like to thank my friends and colleagues at the University of Limoges, for making me feel welcome and for providing me with enjoyable moments.

Lastly, I would like to thank my family for their support and love. Their belief in me has kept me motivated throughout this work.

# Rights

# Table of Contents

# List of Figures

# List of Tables

# Abstract

This thesis focuses on the “simultaneous acquisition of geometry and reflectance”.

RGB-D cameras are used in a variety of applications, including 3D scanning, robot navigation and manipulation, and so on. These cameras provide a color (RGB) image and a depth map simultaneously. The depth map indicates the distance between objects and the camera per pixel. These RGB-D cameras are available in a variety of prices; with the low-cost models intended for general public use. However, these low-cost RGB-D cameras suffer from measurement errors in certain areas / under certain conditions. Therefore, the result depth map contains missing values (holes), incorrect depth measurements, and noise. These issues are most common in areas where objects are transparent, specular, too close or too far away, or too thin.

In this work, we propose an approach for correcting and completing the depth of close-range specular objects. Our approach consists of several steps: First, we create a 3D scene generator that generates a random scene with a table in the center and several objects on it. Each scene is generated by varying different components such as the number of objects, their textures, lighting conditions, and so on. We then use this generator to create a great number of synthetic images for training the neural network. Second, we train neural networks on our synthetic dataset to help identify incorrect regions of the depth map. We then remove these areas in several steps. Eventually, we complete the depth using an optimization method. We test our proposed pipeline on real-world data and demonstrate that it achieves excellent results.

Obtaining an object's material properties (reflectance) from a single image is a challenging and ill-posed task. Previous works employed multiple neural networks and imposed numerous constraints on the input data to obtain these properties. Their imposed constraints include restricting object shapes, not considering any shadows in the images, and so on. Our objective is to define a simple method for estimating an object's reflection properties from a single image that is applicable in everyday use cases, yields acceptable results, and is not overly complicated in design. We propose an approach simplified by an adaptation of the method we proposed previously; we create a new scene generator to generate synthetic datasets and train neural networks to infer reflectance directly.

# Résumé

Cette thèse porte sur « l'acquisition simultanée de la géométrie et de la réflectance ».

Les caméras RGB-D sont utilisées dans une variété d'applications, y compris la numérisation 3D, la navigation et la manipulation de robots, etc. Ces caméras fournissent simultanément une image couleur (RVB) et une carte de profondeur. La carte de profondeur indique la distance entre les objets et la caméra par pixel. Ces caméras RGB-D sont disponibles dans une variété de prix ; avec en particulier des modèles low-cost destinés à un usage grand public. Cependant, ces caméras présentent des défauts de mesure dans certaines zones / sous certaines conditions. Par conséquent, la carte de profondeur obtenue présente des valeurs manquantes (trous), des mesures de profondeur incorrectes et du bruit. Ces problèmes sont plus fréquents dans les zones où les objets sont transparents, spéculaires, trop proches ou trop éloignés, ou trop fins.

Dans ce travail, nous proposons une approche pour corriger et compléter la carte de profondeur d'objets spéculaires proches. Notre approche se compose de différentes parties: Tout d'abord, nous créons un générateur de scène 3D qui génère une scène avec une table au centre et plusieurs objets dessus. Chaque scène est générée en faisant varier différents composants tels que le nombre d'objets, leurs textures, les conditions d'éclairage, etc. Nous utilisons ensuite ce générateur pour créer un grand nombre d'images synthétiques pour entraîner le réseau de neurones. Deuxièmement, nous entraînons des réseaux de neurones sur notre ensemble de données synthétiques pour aider à identifier les régions incorrectes de la carte de profondeur. Nous supprimons ensuite ces zones en plusieurs étapes. Finalement, nous complétons la profondeur en utilisant une méthode d'optimisation. Nous testons notre pipeline proposé sur des données du monde réel et démontrons qu'il obtient d'excellents résultats.

Obtenir la réflectance d'un objet réel à partir d'une seule image est une tâche difficile. Les travaux antérieurs utilisaient plusieurs réseaux de neurones et imposaient de nombreuses contraintes sur les données d'entrée pour obtenir ces propriétés. Leurs contraintes imposées incluent la restriction des formes d'objets, la non-prise en compte des ombres dans l'image, etc. Notre objectif est de définir une méthode simple pour estimer les propriétés de réflexion d'un objet à partir d'une seule image qui soit applicable dans les cas d'utilisation quotidienne, donne des résultats acceptables et ne soit pas trop compliquée dans la conception. Nous proposons une première approche simplifiée par une adaptation de la méthode que nous avons proposée

précédemment : création d'un nouveau générateur de scènes pour générer des ensembles de données synthétiques et entraîner des réseaux de neurones.

# Chapter I. Introduction

## I.1. Motivation

Color images captured by standard 2D digital cameras have been used in many research fields over the last few decades, with great success in computer vision tasks such as image classification, object detection and tracking, semantic segmentation, etc. However, 2D color images lack geometric information and are highly dependent on the illumination conditions of the scene. As a result, solving more complex computer vision tasks that emerge over time with only 2D flat images is challenging.

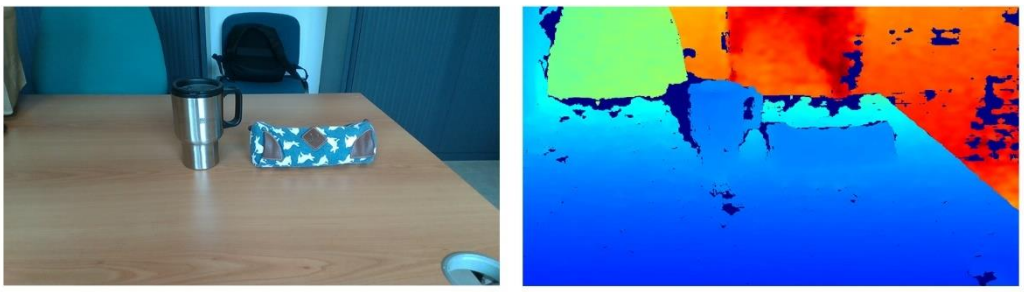

Figure 1: The RGB-D cameras provide a color (RGB) image alongside a depth map. The depth map indicates per pixel how far away objects are from the camera. This data is captured by Intel Realsense D435 camera. The depth map is colorized for better visualization.

In recent years, researchers began to focus on a new type of sensor called RGB-D cameras, since these cameras provide additional information about the scene. The RGB-D cameras provide a color (RGB) image alongside a depth map. The depth map

indicates per pixel how far away objects are from the camera. An example of data captured by an RGB-D camera is illustrated in Figure 1. By using color image and the depth information of a scene together, many new applications and solutions in various fields are introduced. Some of these applications are as following:

- Agriculture: Utilizing agricultural robots is one way to address the shortage of skilled manual laborers and to lower farming costs. These robots use RGB-D cameras to detect and locate crops and fruits and provide useful information about them [FGW20]. They are also capable of harvesting.
- Health: RGB-D cameras can be used in hospitals for a variety of applications. For example, these cameras have been used in operating rooms at Sunnybrook hospital in Canada. Surgeons can then browse and navigate through CT scan images without touching the computer during operations. As a result, they will not need to rescrub [URL10]. Another example is the use of RGB-D cameras in neonatal care departments to assist clinical staff in better supervising and documenting tasks while reducing overload [SGHG21].
- Cultural heritage preservation: Cultural heritage may lose their original shape and color due to passage of time or repeated reparation. It is critical to create a digital model of them for future reference. The digitalization process can be accomplished with RGB-D cameras [GSB18]. Moreover, the digital models can be used for creating online virtual museums.
- Robotic assistance: RGB-D cameras can assist robot grasping and manipulation tasks such as dishwashing, sorting and cleaning objects, and so on [SMPN20].
- 3D Printing tasks: In recent years, many affordable 3D printers have become available for general public use. However, few people are expert in 3D modelling. RGB-D cameras can be used to scan objects and then re-print them in different sizes using 3D printers.

While these cameras provide useful information, they differ in terms of price and acquisition quality. High-quality RGB-D cameras are typically very expensive and not available for general public use. For example, Matterport Pro 3 is a high-quality sensor that captures accurate and detailed 3D data, but costs over 5000 euros [URL8].

The first low-cost RGB-D camera was introduced by Microsoft in 2010. The company introduced Kinect v1 for gaming purposes with the Xbox console. The sensor can track human body and hand motion without the use of any controller (Figure 2). Later, many more low-cost cameras were introduced such as Microsoft Kinect v2, Intel

Realsense D435 [IR22], etc. Furthermore, many manufacturers have recently begun to incorporate RGB-D cameras into a variety of products, including laptops and smartphones.

Even though low-cost RGB-D cameras are more accessible to the general public, they have some limitations; The result depth map contains missing values (holes) or incorrect measurements and noise because these cameras are unable to accurately measure how far away certain areas are. These issues typically occur in areas where the objects are transparent or specular, too near or too far away, or too thin. There has been a lot of research on how to complete the depth map of these affordable cameras in indoor and outdoor scenes, but not so much on close-range objects. In this work we propose an algorithm to correct and complete the depth of close-range specular objects. We enhance the depth map's quality by benefiting from the corresponding 2D color image and the advances in the data driven approaches.


Figure 2: Microsoft introduced Kinect sensors for gaming purposes with the Xbox console. The sensor can track human body and hand motion without the use of any controller. (Image from [URL9])

Another area of research is the estimation of material properties from real-world objects. 3D artists generally try to achieve realistic rendering by using material properties that are similar to those of real-world objects. There are numerous analytic reflection models based on physics for modeling these properties. However, these models have multiple parameters, and it is time-consuming to try to guess them by only observing the real-world objects and choosing different values in 3D modelling

software. Another way to estimate the material properties is using gonioreflectometer. This device produces very accurate results, but it is very expensive, and the process requires a lot of time. In this work we discuss a new approach of acquiring material properties of an object from a color image while taking advantage of deep learning advancements. As a result, RGB-D cameras would be able to simultaneously retrieve an object's material properties while capturing its depth. Since our proposed approach only relies on the color image, it is also possible to apply it to the mobile phones equipped with simple RGB cameras.

## I.2. Contribution and Outline

This work is arranged as follows:

Chapter I. Introduction:

In chapter one, we provide an overview of this work. As we outline the context, we then describe the problem that exists, the purpose of the thesis, and its application to real-life situations.

Chapter II. Fundamental and Previous work:

In chapter two, we present the fundamentals and past work related to our research. we begin by discussing the technologies used in RGB-D cameras to measure depth. Then, we present the fundamentals of deep learning as well as the past work in two different computer vision tasks: Image classification and semantic segmentation. We investigate previous network architecture designs and discuss the overall comparison of them. Following that, we discuss the state-of-the-art in three additional tasks related to our research: 3D scene understanding from single image, depth map completion and reflectance estimation. The study of these previous works allows us to focus on the best approaches for our research.

Chapter III. Depth Completion for Close-Range Specular Objects:

In chapter three, we present an approach to correct and complete the depth map of close-range specular objects using neural networks and optimization algorithm. We explain thoroughly in different sections, the various parts of the approach such as dataset generation, network architecture and training, incorrect depth removal and the final depth completion. We then discuss the results of this approach on the depth map of the Intel Realsense D435 camera, as well as its limitations and future work. We presented this approach as a full article in WSCG 2022.

Chapter IV. Reflectance Estimation:

In chapter four, we discuss our approach to estimate the material properties of an object form color image using neural network. We perform various experiments on the real and synthetic data to evaluate the method.

Chapter V. Conclusion:

In this chapter, we provide a summary of our work followed by a general conclusion. Furthermore, we describe the limitations of our approach and suggest future research directions.

# Chapter II. Fundamental and Previous work

In this chapter we discuss the fundamental and previous studies related to our work. In the first section, we explain what depth cameras are and then discuss their various types. In the second section, we discuss the fundamentals and the state of the art in deep neural network architecture design. In the third section, we talk about the recent approaches of estimating underlying properties of a single color image. Eventually at the last section, we go over the most recent methods for completing the depth map of indoor scenes.

## II.1. RGB-D Cameras

The commodity RGB-D cameras capture two types of data simultaneously: an RGB image which is the typical color image and a depth map. A depth map is a pixel-map that provides the distance from the camera to the objects in the scene at each pixel. These two data are usually captured through separate lenses that are located apart from each other.

There are two types of RGB-D cameras: passive sensors and active sensors. Passive sensors require external illumination, whereas active sensors have projectors that project light onto the scene. All technologies used by RGB-D cameras are illustrated in Figure 3.

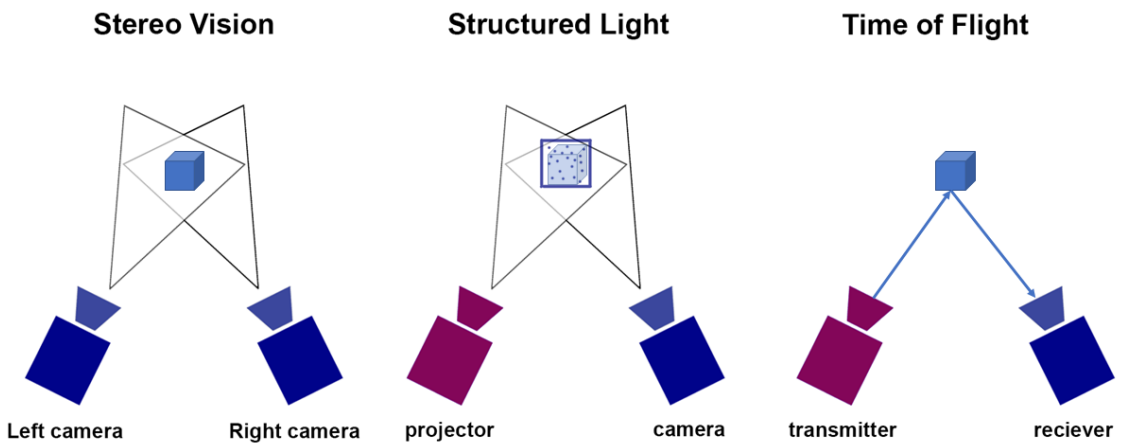


Figure 3: Stereovision, Structured Light, and Time of Flight are common technologies used in RGB-D cameras for depth sensing.

Passive sensors use stereo vision technology which is based on binocular human vision. They use two (or more) lenses to capture images of a scene from different viewpoints, and then use the triangulation method to calculate the depth. These sensors perform well under natural illumination and direct sunlight. However, they struggle in low-light conditions, texture-less scenes, or occlusion situations, in which one object is visible through one lens but not the other. Also, their depth sensing technique is computationally expensive since the corresponding points need to be founded in different images (from different viewpoints) of a scene. One example of this type of sensor is TaraXL camera from e-con Systems™ [URL7]. It is shown in Figure 4.

Active sensors are categorized into two groups: Structured light and Time of flight (ToF).

- Structured light sensors contain an infrared projector and a camera. The IR projector emits a pattern onto the scene which deforms according to the geometric shape of the objects present in the scene. The camera determines the depth by observing the pattern deformations. One of the popular cameras in this category is Microsoft Kinect V1 which is illustrated in Figure 5. The advantage of these types of cameras compared to passive sensors is that they project a pattern into the environment, which enables them to perform well in texture-less scenes. They do, however, have some drawbacks; For instance, they are only suitable for indoor use because they perform poorly under direct sunlight. Moreover, they struggle to work correctly on shiny or dark objects.

Figure 4: TaraXL camera from e-con Systems™ [URL7] uses passive stereo technique to capture the depth. It uses two lenses to capture images of a scene from different viewpoints, and then uses the triangulation method to calculate the depth.

- Time of Flight sensors contain a transmitter and a receiver. In this type of sensor, the depth is measured by the amount of time it takes for a signal to travel from the transmitter to the scene, be reflected, and then be received back by the receiver. Microsoft Kinect V2 sensor is in this category (Figure 5). Time-of-Flight sensors can operate in one of two ways: direct (pulsed ToF) or indirect (modulated ToF). The pulsed ToF sensor emits pulses of light and measures the time they return, whereas modulated ToF sensor emits a continuous modulated light and measures the phase shift of the returned signal. These types of sensors have similar disadvantages as structured light sensors, that is, they perform poorly in the presence of strong sunlight and on shiny or dark surfaces.


**Figure 1: Microsoft Kinect v1** **Figure 2: Microsoft Kinect v2**

Figure 5: Illustration of two common RGB-D cameras: Microsoft Kinect v1 and Microsoft Kinect v2. Microsoft Kinect v1 uses structured light technology and has an infrared projector. Microsoft Kinect v2 uses Time-of-flight technology and contains an emitter and a receiver for depth sensing. Figure from [PBN16].

In recent years, newer cameras have combined active and passive technologies to achieve improved performance. Intel, for example, introduced D435 and D415 sensors as part of the D400 series [IR22], which employ IR active stereoscopy technology. These two sensors use stereo vision along with an infrared projector, which improves the camera performance by projecting a pattern onto the scene. As a result, they can capture depth in texture-less and dark scenes. The D435 and D415 sensors are illustrated in Figure 6.


Figure 6 : Intel Realsense D415 (left) and D435 (right). They both use stereo vision with an infrared projector. Figure from [URL2]

Nevertheless, the quality of depth map captured by low-cost RGB-D cameras also depend on the material properties of the objects present in the scene. In chapter 3, we first study the limitations of Intel Realsense D435 camera in details before proposing an approach to correct the measured depth using deep learning.

## II.2. Deep Learning

The field of computer vision has become increasingly important over the last few decades. In computer vision, researchers try to create systems that process and comprehend images. One approach is using data-driven methods which are inspired by the human visual system. The idea is that even people comprehend better the images they see by relating them to previous experiences. As a result, in these data-driven approaches, the computer learns from a large amount of data and interpret the new data using its prior knowledge. In this section, we discuss the recent data-driven approaches for two different tasks in computer vision: image classification and semantic segmentation. Figure 7 illustrates these two tasks. In image classification, the computer attempts to understand the image as a whole while in segmentation, the computer understands the image in pixel level and divides each image to different segments.

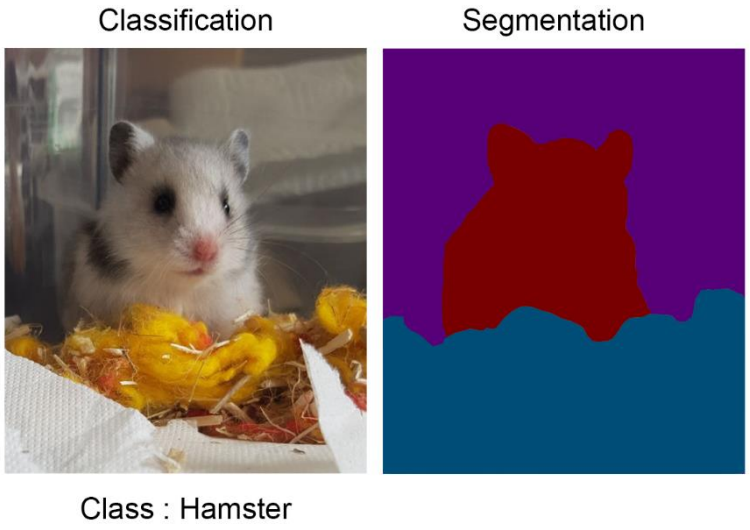


Figure 7 : In classification, the computer understand the image and assigns it to a class (label). In segmentation, the computer understands the image in pixel level and divides each image to different segments.

## II.2.1. Image Classification

In image classification, the algorithm tries to understand an image globally and assign it to a known category (class). In this task, the number of categories is fixed. Image classification is one of the fundamental components of computer vision and is used in more complex tasks like object detection (where the algorithm tries to find the exact location of an object in an image), semantic segmentation (where the algorithm attempts to assign a label to each pixel) and image captioning (where the algorithm tries to describe an image with a sentence), etc.

One way of performing image classification, is using convolutional neural networks. These networks consist of several components:

- Convolutional layers: Convolutional layers are specifically designed to process images because they preserve the spatial relations of the pixels. Each convolutional layer takes a four-dimensional tensor as input (a batch of images/feature maps), convolves it with various weight matrices (also known as filters/kernels), and outputs a four-dimensional tensor (a batch of feature maps).

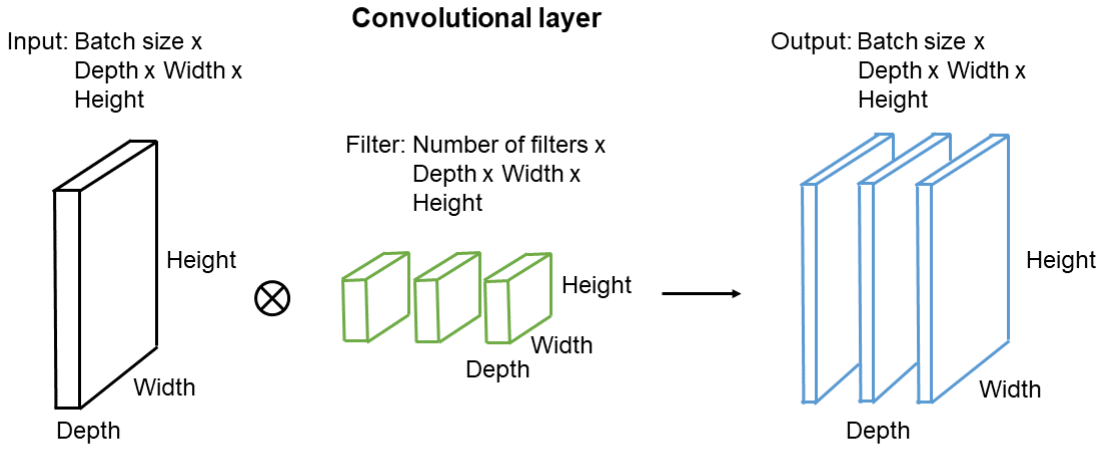


Figure 8: Each convolutional layer takes a four-dimensional tensor as input (a batch of images/feature maps), convolves it with weight matrices (also known as filters), and outputs a four-dimensional tensor (a batch of feature maps).

In general, each filter has the same depth as the depth of the input tensor (also known as channel). When it comes to color images, the input tensor has a depth of three, which is R, G, and B channels. The depth of the output tensor is equal to the number of filters in that layer. These convolutional layers can be stacked on top of one

another in the networks with activation functions in between them (different types of activation functions are introduced in a later section).

During the convolution operation the filters are slid across the entire spatial positions of the input tensor, and the dot product of the filters and that section of the input tensor is computed. Figure 9 shows an example of the operation. For simplicity, here the input is a single feature map with a depth of one and width and height of 5x5. The filter has the depth of one and the width and height of 3x3. The output feature map is illustrated in blue. The filter slides over the input feature map and performs dot product to produce the output. “Stride” describes how many pixels the filter moves each time, which is one in this example. The parameters of each filter are learned throughout the training. If the spatial dimension of a squared input is n and the spatial dimension of a squared filter is k, and the stride is one, the spatial dimension of the output is calculated as:

$$output_{dim} = n - k + 1$$

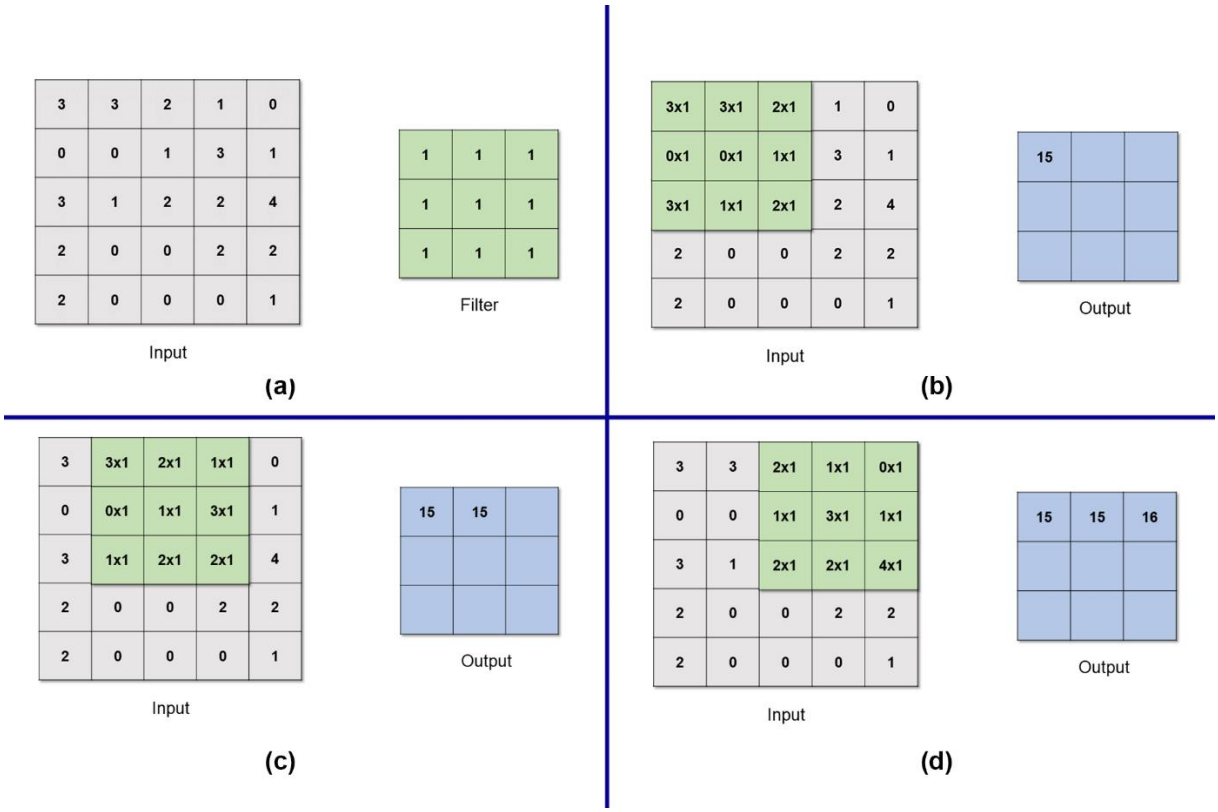


Figure 9: During the convolution operation, the filter slid across the entire spatial positions of the input tensor, and the dot product of the filter and that section of the input tensor is computed. In this example (b-d), the filter moves by one pixel at each step (stride = 1).

However, as seen in Figure 8, the spatial dimension of the feature map is decreased with each convolutional step (even with stride of one). Therefore, it would be challenging to construct deep CNNs. The solution is to extend each input feature map by adding extra pixels on the outer dimensions of it. In neural network terminology, this is referred to as "padding". Figure 10 shows an example of padding = 1. It is possible to choose the new added pixel values in various ways; for instance, they can be zero (as in this example – also known as zero-padding), or they can be the same value as the closest pixel at the border. If we refer to the padding as p, the spatial dimension of the output with stride of one can now be calculated as:

$$output_{dim} = n + 2p - k + 1$$

Or more generally with the stride s:

$$output_{dim} = [(n + 2p - k)/s] + 1$$

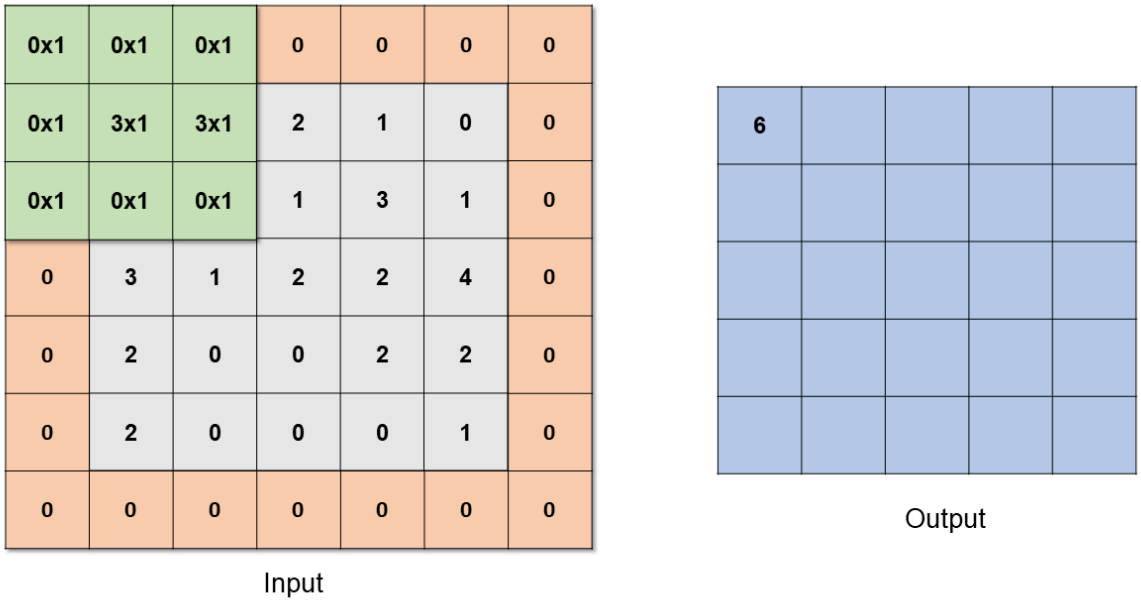


Figure 10: Padding is used in CNNs to control the amount of spatial information lost during convolutions. In this example we used padding of one with pixel values of zero (zero-padding - illustrated in pink).

- Pooling layers: These layers are similar to the convolutional layers, but their main objective is to reduce the spatial dimensions of the input (down sampling). For each pooling layer, a filter size and a stride are defined (no padding).

However, rather than performing the dot product, a fixed function is used on the input to summarize the values. Therefore, pooling layers do not contain any learnable parameters. Here are two of the most commonly used fixed functions for pooling:

1. Max pooling
2. Average pooling

Figure 11 illustrates an example of non-overlapping max pooling.

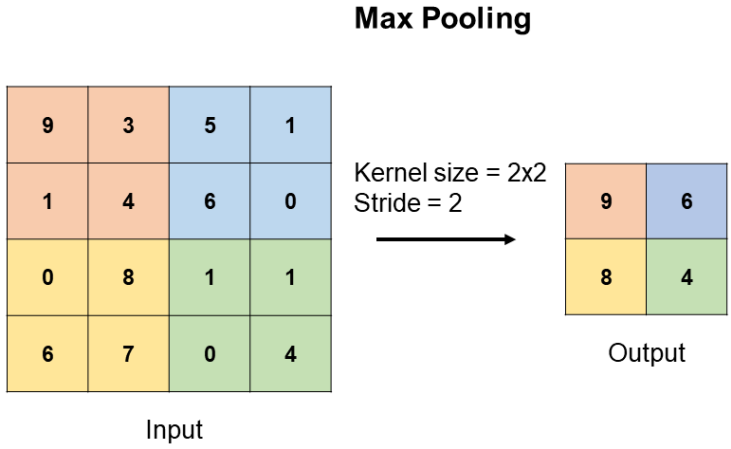


Figure 11: Here is an example of Max pooling which performs max function on 2x2 regions of the input. The main goal of a pooling layer is to downsample the input.

- Activation functions: Activation functions are one of the most critical components of the neural networks. They are used to add non-linearity to the network. There are numerous types of activation functions available, some of which are demonstrated in Figure 12.

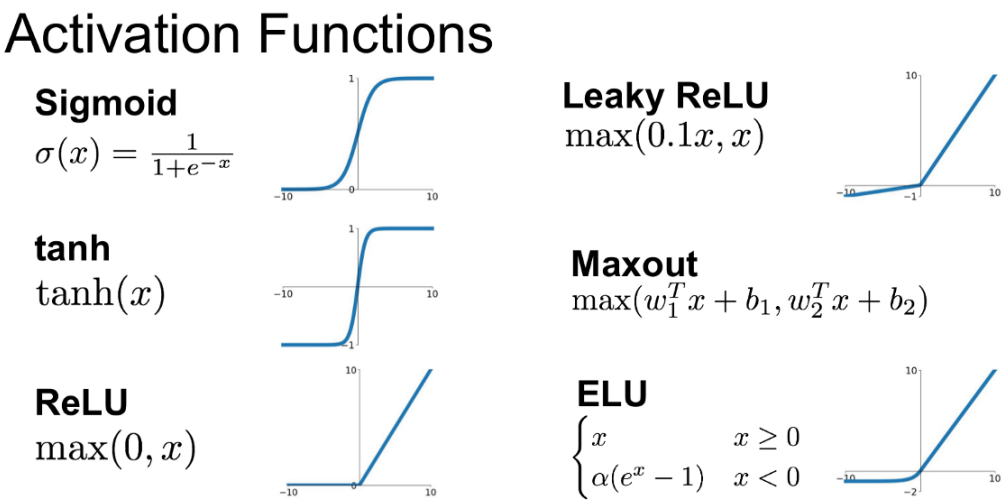


Figure 12: Activation functions add non-linearity to the neural network. Some of the most common activation functions are illustrated in this figure.

- Fully-connected layers: The fully connected layers consist of three types of layers: input layer, hidden layers and the output layer. The size of the input and output layers is fixed. Figure 13 illustrates a fully connected network with input and output size of three and two hidden layers. Each layer consists of many nodes that are connected to the nodes of the next and previous layers. Each connection has a weight that is learned through the training process. At each node, every input value is multiplied by the corresponding weight, and then all values are summed together. Next, a bias value is added to the sum, and the result goes through an activation function. Here are the equations for each node:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

$$output = g(z)$$

Where n is the number of inputs, x is the input value, w is the corresponding weight, b is the bias, and g is the activation function.

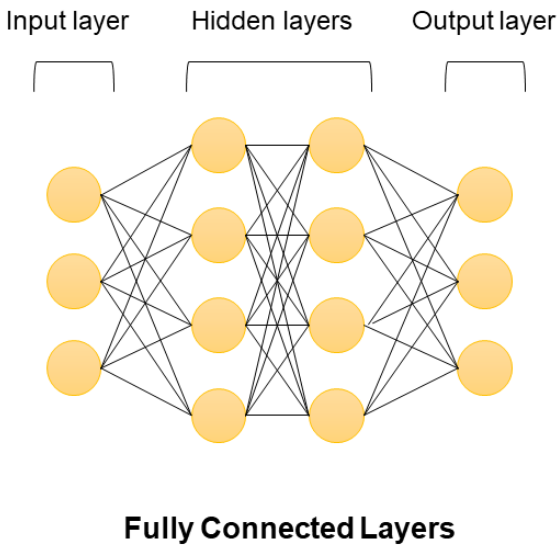


Figure 13: The fully connected layers comprised of three types of layers: input layer, hidden layer(s) and the output layer. Each layer consists of many nodes that are connected to the nodes of the next and previous layers.

- Normalization layers: The most common used normalization in CNNs is “batch normalization”. These layers are generally used after convolutional/fully-connected layers and before the activation functions. Their main objective is to speed up the training of the network. For more information about batch normalization please refer to the original paper [IS15].

Nevertheless, it is challenging to figure out how to choose and combine these components to achieve the best results. In the following section, we discuss the previous work on designing convolutional neural network architecture for image classification.

### II.2.1.1. Convolutional Neural Network Architecture for Image Classification

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [RDSK15] was a competition that took place yearly between the years of 2010 until 2017. The contest was about creating the best model to perform image classification and object detection on ImageNet dataset [DDSLLF09]. This contest led to significant amount of research and advancements in the design of convolutional neural network architecture. Neural networks were used for the first time in 2012, and prior to that, the winners did not use any deep learning approach. In 2012, Krizhevsky et al. [KSH12] won the classification

and localisation competition by proposing a deep convolutional network model named AlexNet. AlexNet has 8 layers in total: 5 convolutional layers combined with max-pooling layers, and 3 fully-connected layers at the end of the network. The network uses ReLU activation function. Figure 14 shows the overall schema of the AlexNet.

Due to the GPUs' low memory capacity during that time, the entire architecture was distributed and trained across two GPUs.

When designing AlexNet, all kernel sizes were hyperparameters and the entire network architecture was designed through trial and error. Thus, it was not easy to extend the network. Over time, researchers tried to find some general rules to design the networks that would reduce the experiments and hyper-parameter tuning process.

In 2014, Simonyan and Zisserman designed VGG architecture [SZ14]. This architecture follows specific design rules:

- The convolutional layers have the kernel size of 3x3 and the stride and padding of 1 pixel.
- All the pooling operations are performed by max-pooling and have the size of 2x2 with the stride of 2 pixels.
- The number of channels doubles after each max-pooling layer.
- Following the convolution layers, there are fully-connected layers with the same configurations as AlexNet.
- The network uses ReLU activation function.

As a result of these design rules, researchers would be able to lengthen the networks without spending too much time on experimenting the size and type of CNN components. The VGG architecture has two most used variations: VGG-16 and VGG-19, where the number 16 and 19 indicates the total number of layers. Overall schema of VGG-16 and VGG-19 is illustrated in Figure 14.

Until then, the tendency was to build larger networks in order to get better performance; this means building a deeper network with more units at each level. However, larger networks require a larger training dataset and more computational resources to learn effectively. Szegedy et al. [SLJS14] aimed to build a network that performs well without being computationally complex; Therefore, they proposed GoogLeNet. GoogleNet is a 22-layer network with the following features: the network starts with a module that quickly downsamples the input image to avoid expensive convolutions on large spatial feature maps. Afterward, the entire network is represented by repeating a local structure known as the "inception module."

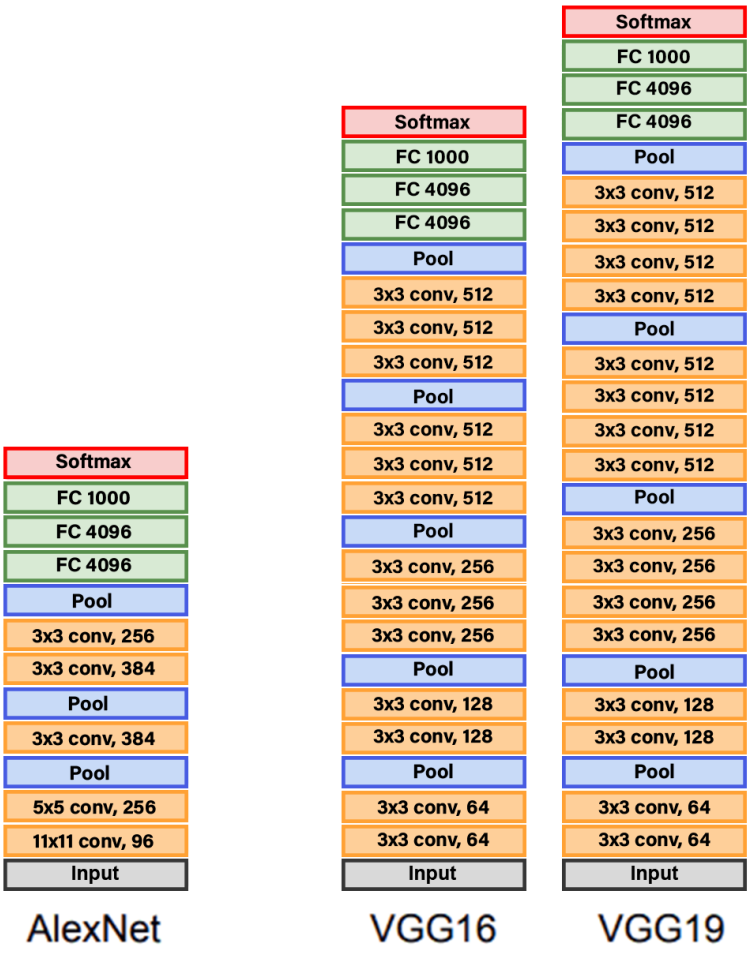


Figure 14: Overall schema of AlexNet, VGG-16 and VGG-19. AlexNet has 8 layers in total: 5 convolutional layers (with various kernel sizes) combined with max-pooling layers, and 3 fully-connected layers (FC) at the end of the network. In VGG architecture design, all convolutional layers have the same kernel size of 3x3, with max-pooling of 2x2, and the number of channels doubles after each layer of max-pooling.The FC layers of VGG are the same as AlexNet. (Figure from Stanford lectures [URL1])

The inception module uses parallel computation to perform convolutions with kernels of various sizes (1x1, 3x3, and 5x5) and a max pooling simultaneously. However, before performing the 3x3 and 5x5 convolutions, it performs a 1x1 convolution to compress the input. In the end, the network consists of an average pooling layer followed by one fully-connected layer that maps the features to the 1000 classes of ImageNet. Using average pooling rather than the numerous fully-connected layers allowed for the reduction of a large number of parameters. The entire network is illustrated in Figure 15.

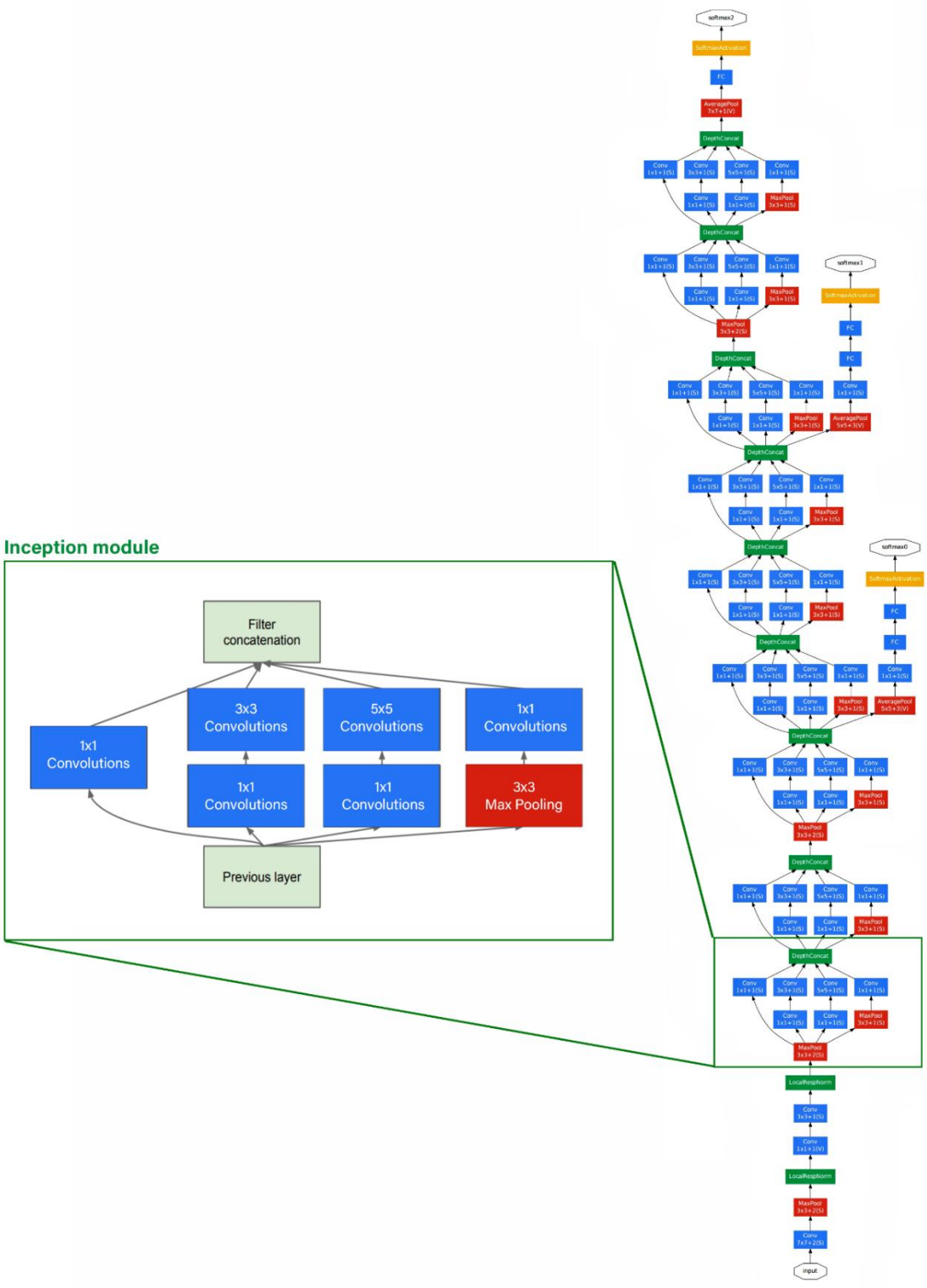


Figure 15: Entire GoogLeNet architecture (Figure from [SLJS14]); The inception module is repeated throughout the network. This module uses parallel computation to perform convolutions with kernels of various sizes (1x1, 3x3, and 5x5) and a max pooling simultaneously. However, before performing the 3x3 and 5x5 convolutions, it performs a 1x1 convolution to compress the input.

In 2015, He et al. [HZRS16] experimented with larger networks and noticed that as the networks get deeper, they perform worse than the shallower ones because they are harder to train and optimize. However, they stated that if the layers from a shallow network were copied to the layers of a deeper network and the remaining layers of the deeper network only performed identity mapping, the deeper network should be able to perform at least as well as the shallower one. Thus, they introduced residual blocks and created the Resnet architecture. The residual blocks add the input to the output of each block as illustrated in Figure 16.

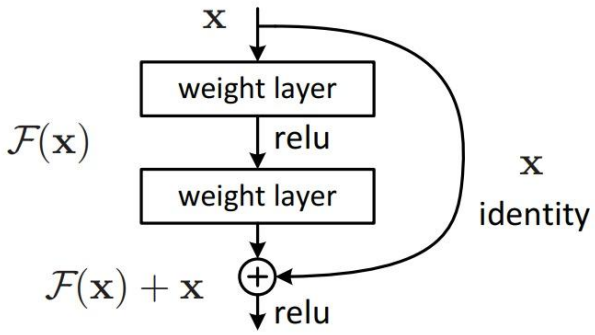


Figure 16 : Residual Block adds the input to the output (Figure from [HZRS16])

Resnet architecture contains convolutions with kernel sizes that are mostly 3x3, which is inspired by the VGG architecture [SZ14], and instead of having many fully-connected layers at the end, it has average pooling with one fully-connected layer, similar to GoogLeNet [SLJS14]. The residual blocks are repeated throughout the network as illustrated in Figure 17. The residual block and a slightly different version called “Bottleneck” block enabled He et al. to design networks of various sizes, deeper than previously built, such as Resnet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. ResNet-152 is 8 times deeper than VGG-19 but has less computational complexity. The ResNet architecture design was a huge success in neural network architecture history, winning numerous competitions that year.

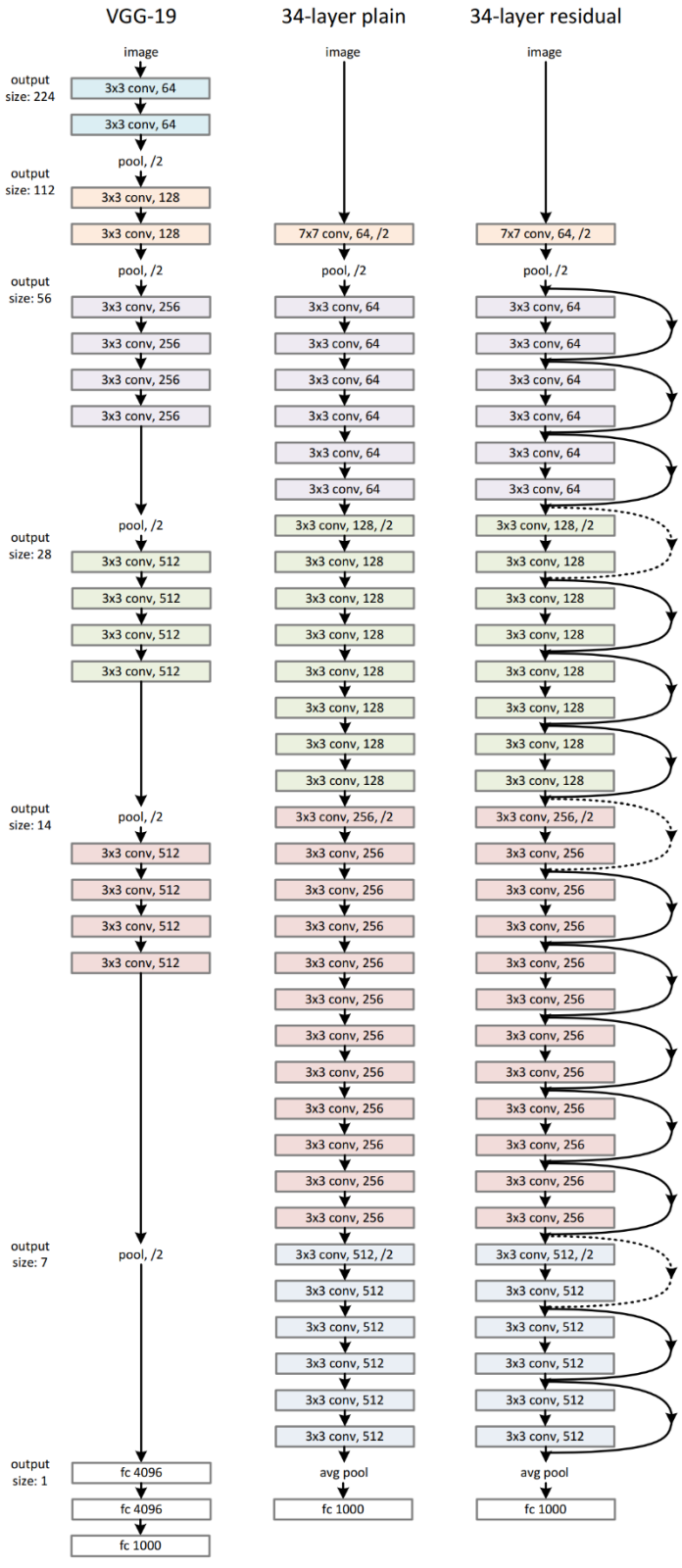


Figure 17 : Illustration of ResNet-34 vs VGG-19, both designed for classification on ImageNet dataset (Figure from [HZRS16]). Resnet architecture contains convolutions with mostly 3x3 kernel sizes, which is inspired by the VGG architecture. The residual blocks are repeated throughout the network.

Szegedy et al [SVISW16] proposed some modifications to GoogLeNet to reduce the computational cost and they proposed Inception-v2 and Inception-v3. The modifications include some adjustments to the convolutional filter sizes (e.g., replacing 5x5 kernels with two 3x3 kernels, or using asymmetric kernels like nx1), introducing a new regularization method called “label smoothing”, etc.

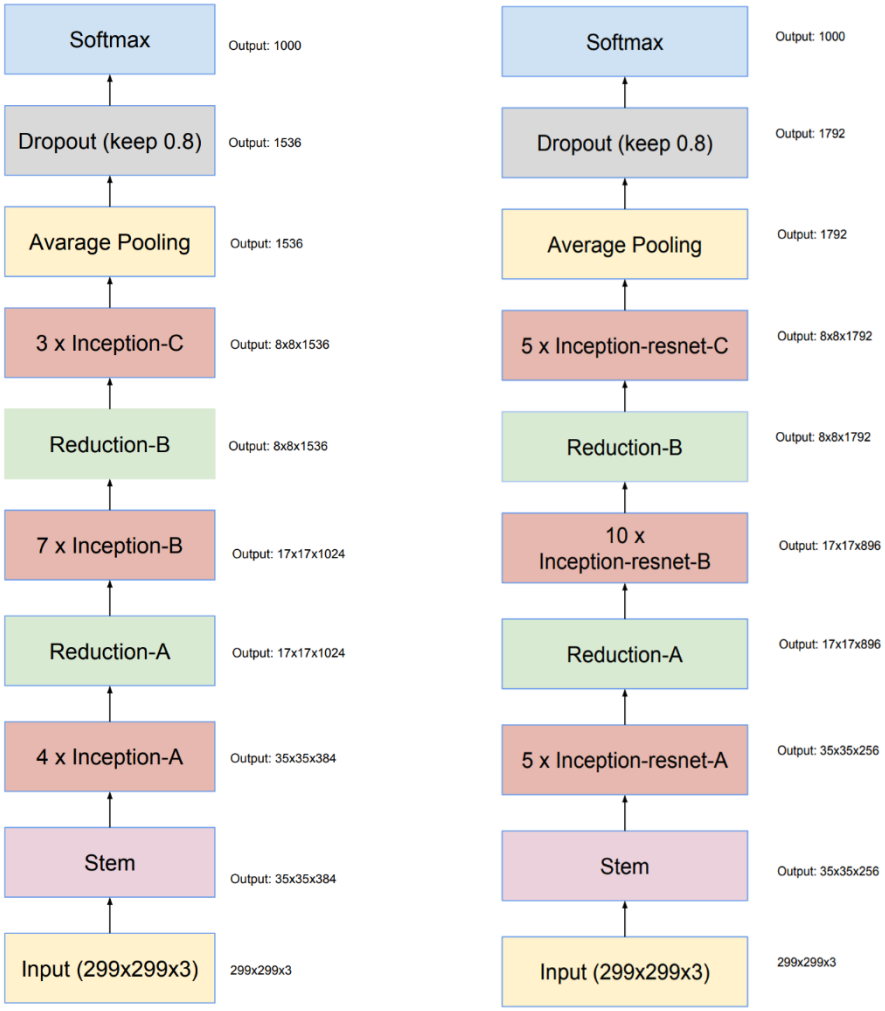


Figure 18: Entire inception-v4 (left), Inception-ResNet-v1 and Inception-ResNet-v2 (right) model architectures. The Inception-ResNet-v1 and Inception-ResNet-v2 models have the same architecture but their underlying modules differ. For more information on underlying modules (Inception(resnet)-A, Inception(resnet)-B, Inception(resnet)-C and Reductions) please refer to the original paper. (Figure from [SIVA16])

Later Szegedy et al [SIVA16] proposed three model architectures to further improve the inception models accuracy: Inception-v4, Inception-ResNet-v1 and Inception-ResNet-v2. Compared to Inception-v3, Inception-v4 has a larger number of inception modules, but the architecture is simpler and more uniform. Considering the great success of ResNet architectures [HZRS16], Szegedy et al also attempted to combine the residual connections with the inception architecture and create the Inception-ResNet model. The overall model architectures are illustrated in Figure 18. Their experiments demonstrate that the inception-v4 and Inception-ResNet-v2 models perform similarly, but better than inception-v3 and Inception-ResNet-v1. Also, the Inception-ResNet models converge slightly faster than pure inception models.

Since then, many new architectures have emerged on a frequent basis, most of which are inspired by previous works. Bianco et al. [BCCN18] provided a throughout analysis and comparison of more than 40 neural networks in terms of model and computational complexity, accuracy, memory consumption, etc. They conducted the experiments on two different platforms: a workstation that used an NVIDIA Titan X Pascal GPU and an NVIDIA Jetson TX1 embedded system. They measured the accuracy on the ImageNet dataset for classification task. Figure 19 illustrates a comparison between the neural networks in terms of computational complexity, accuracy, and model complexity. The x axis shows the computational complexity (floating point operations of a forward pass), and the y axis shows the accuracy. The size of the circle represents the complexity of the model which is the number of learnable parameters in total.

As we can see, AlexNet has low amount of computational complexity, low accuracy, and moderate number of learnable parameters. GoogLeNet has less parameters, but the accuracy is still poor compared to most of the architectures. The VGG_BN are the VGG architectures that are trained using the batch normalization technique.

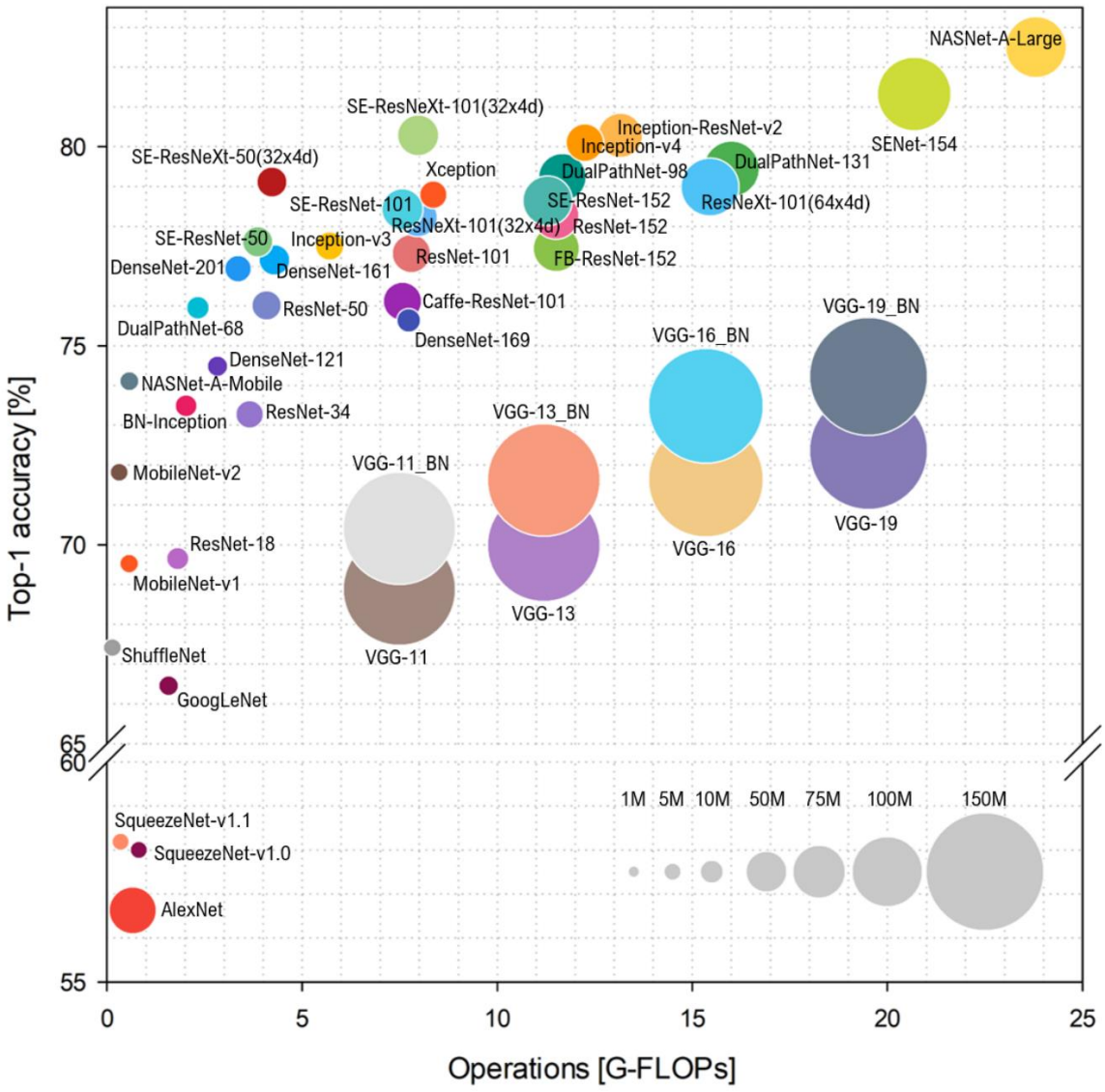


Figure 19 : Comparison of more than 40 neural networks [BCCN18]; The x axis shows the computational complexity (floating point operations of a forward pass) and the y axis shows the accuracy. The size of the circle represents the complexity of the model which is the number of learnable parameters in total. The result is measured on both workstation and the embedded board.

Despite the fact that all VGG architectures have very large number of learnable parameters, they are not very accurate which makes them inefficient. On the other hand, ResNet and Inception architectures have smaller number of parameters but have high accuracy results.

## II.2.2. Semantic Segmentation

Semantic segmentation is another important task in computer vision that is widely used in different domains such as diagnosing disease in medical images, autonomous driving, separating foreground from background in photographs, etc.

In contrast to image classification, which assigns a label to the entire image, semantic segmentation takes an image as input and assigns a label to each pixel. In image classification, the network starts with convolutional layers which are then followed by fully connected layers (e.g VGG [SZ14]) or average pooling (e.g GoogLeNet [SLJS14]). However, in these cases, the size of the input image to the network would be fixed because of the fully connected layers. The first convolutional layers of image classification architectures can be referred to as "encoders" because they encode the input image by decreasing its spatial size and increasing its dimensions.

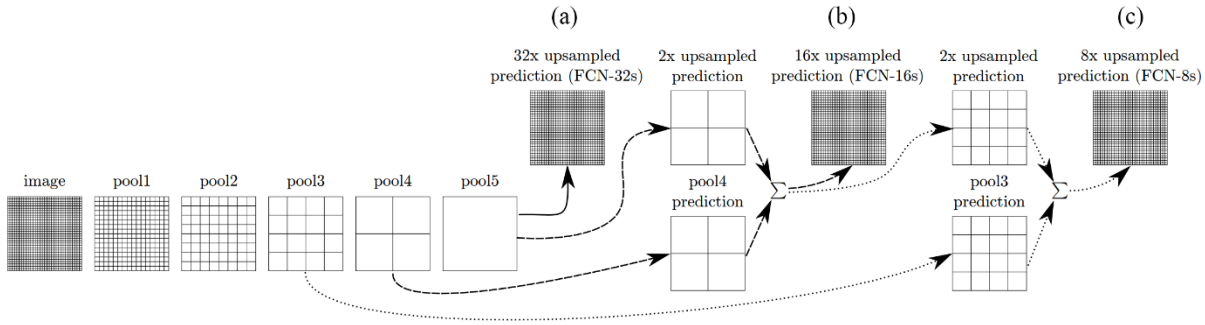


Figure 20: Overall workflow of FCN [LSD15] encoder-decoder.

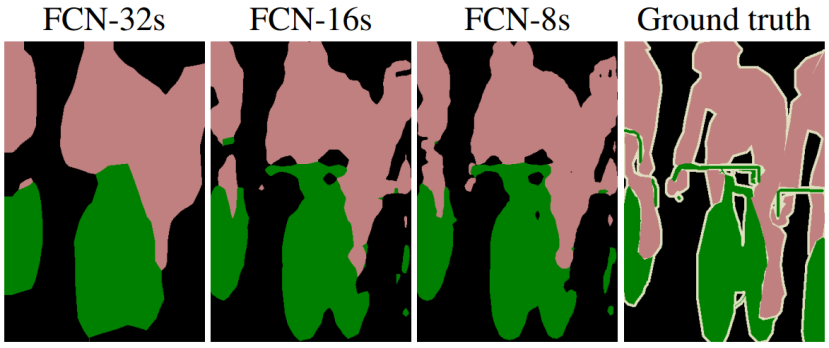


Figure 21: Segmentation results of FCN [LSD15]. It is important to note that adding features from lower layers to the up sampling layers adds more details to the final segmentation result.

In 2015 Long et al. [LSD15] proposed fully convolutional network (FCN) for semantic segmentation. Their network contains only convolutional layers, so there is no restriction on the size of the input image, and the network can receive any arbitrary sized input. The network consists of an encoder (also known as backbone) and a decoder. The encoder first down samples the input to generate a feature map and later the decoder up samples it to the original size using deconvolution. An overview of FCN's workflow can be seen in Figure 20. First, the input image is down sampled, then it up sampled in 3 different stages: in (a) the network directly up samples the feature map to the original size, in (b) and (c) the network up samples the feature map in steps and adds the features in the down sampling layer to it, to get finer details. The results from these steps are illustrated in Figure 21. It is important to note that adding information from lower layers to the up sampling layers adds more details to the final segmentation result.

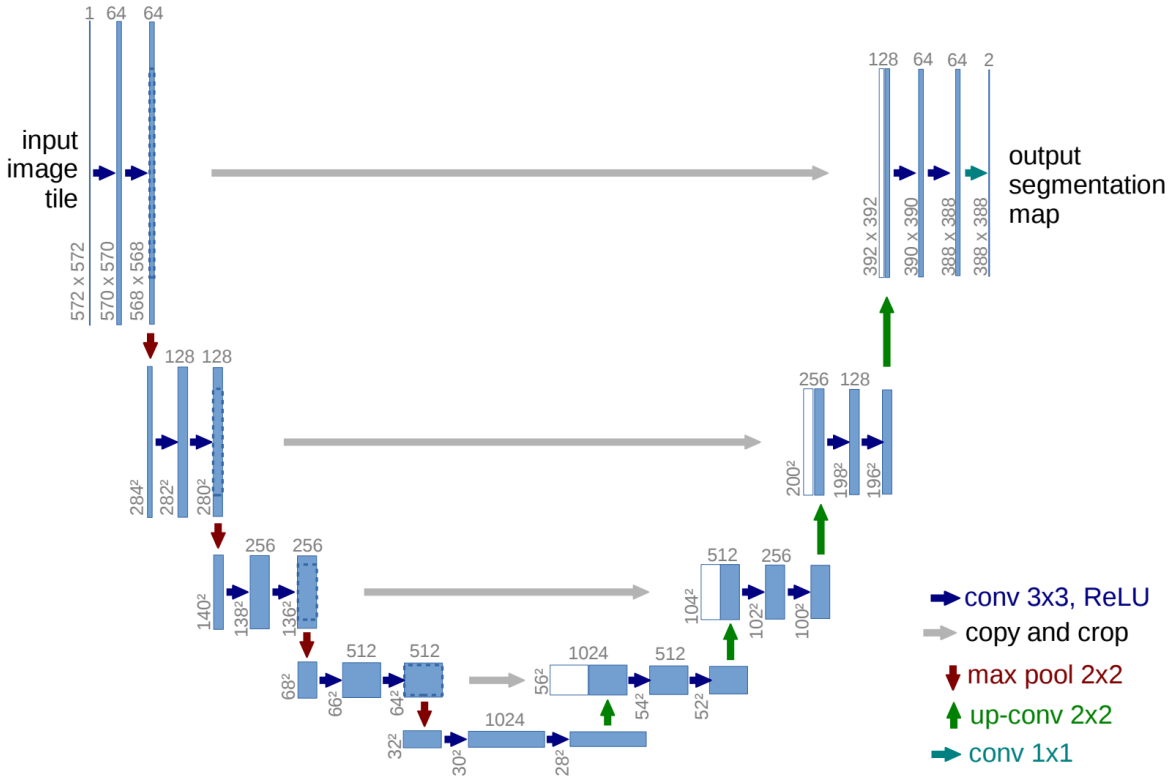


Figure 22: U-Net architecture proposed by Ronneberger et al. [RFB15]. The down sampling and up sampling layers are symmetric. The corresponding layers of down sampling and up sampling are connected using skip-connections.

Ronneberger et al. [RFB15] modified the FCN architecture and proposed the U-Net architecture, which they then used in biomedical segmentation. U-Net is an encoder-decoder network that has a u-shape architecture; the down sampling and up sampling layers are symmetric. The entire architecture is illustrated in Figure 22. The corresponding layers in down sampling and up sampling are connected using skip-connections. U-Net architecture produced accurate segmentation results while requiring less training data.

Chen et al [CPKMY17] discussed different problems relating to the segmentation tasks in their study and proposed a new approach called DeepLabV2. They proposed three new concepts in their design, which are as follows:

- They proposed using a different type of convolution known as *atrous convolutions* to increase the field of view without increasing the number of parameters or computational cost. When compared to standard convolutions, the *atrous convolution* produces a higher resolution feature map. This technique has been widely used in signal processing tasks [HKMT90].

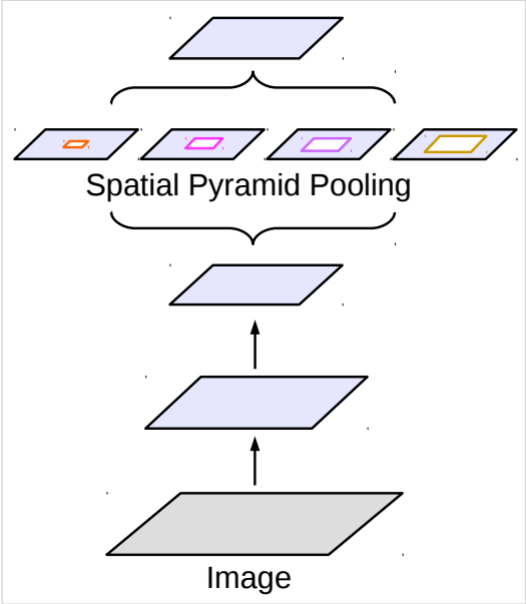


Figure 23: The ASPP module (*atrous spatial pyramid pooling*), uses parallel computing of different *atrous convolutions* and concatenate the result at the end. (Image from [CPSA17])

- Since each image may contain objects of the same class but in different sizes, they proposed a module called ASPP (*atrous spatial pyramid pooling*) for improving the segmentation result. This module uses parallel computing of different *atrous convolutions* (similar to the idea of inception module previously discussed [SLJS14])
- As previously discussed, the output of a fully convolutional network for the segmentation tasks may not capture the details. One way to improve is using skip-connections as Long et al. proposed in FCN [LSD15] (Figure 21). Chen at al proposed an alternative approach using a fully connected CRF (Conditional Random Field) [KK11].

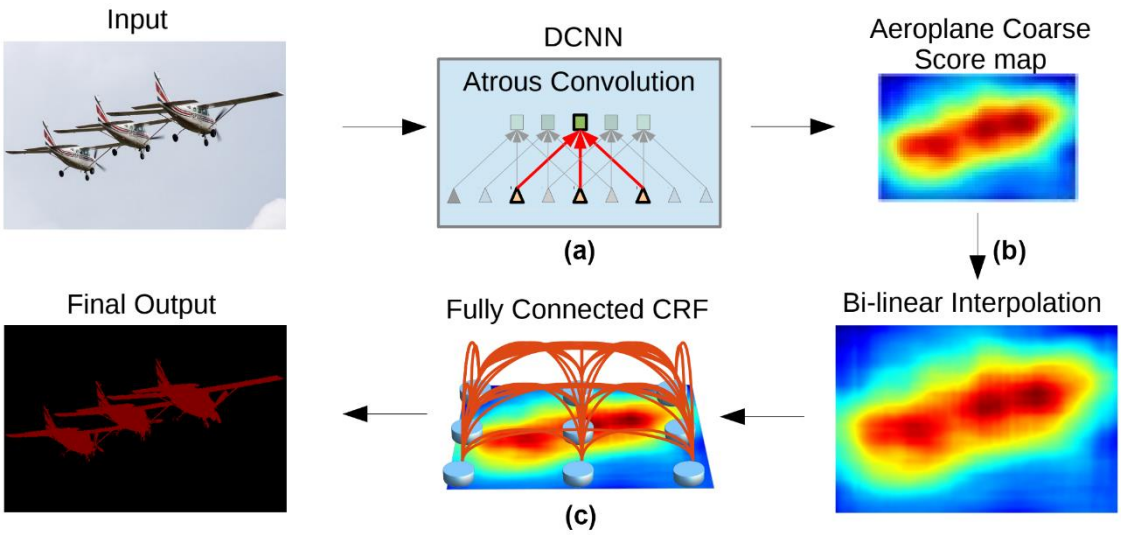


Figure 24 illustrates the overall pipeline of the DeeplabV2 [CPKMY17]. Their pipeline manly consists of a deep CNN (DCNN) that perform the segmentation task and a CRF that refines the result. (a) First the network takes an image as input and performs convolutions using the atrous convolutions. At the end of the convolutions there is the ASPP module that produces the output. (b) Next, the result is upsampled using bilinear interpolation to reach the resolution of the input image and eventually (c) it goes through a fully connected CRF for refinement.

Their full pipeline is illustrated in Figure 24. Their pipeline manly consists of a deep CNN (DCNN) that perform the segmentation task and a CRF that refines the result. First the network, which is based on a VGG-16 [SZ14] or ResNet-101 [HZRS16], takes an image as input and performs convolutions using the *atrous convolutions*. At the end

of the convolutions there is the ASPP module that produces the output. Next the result is upsampled using bilinear interpolation to reach the resolution of the input image and eventually it goes through a fully connected CRF for refinement.

Later, Chen et al proposed DeepLabV3 [CPSA17]. In their approach, they modified the DeeplabV2 concept by removing the Fully Connected CRF and only performing the segmentation using CNN and the ASPP module. They also modified the ASPP by adding batch normalization and global average pooling.

## II.3. Estimating Underlying Properties of an Image

The challenge of producing realistic images in computer graphics has been well studied throughout history. This process requires a rendering equation to be solved to obtain each pixel's color in the image plane of the camera. With technological advancements in recent years, new tasks such as augmented reality have emerged that require inverting this process and determining the underlying components of images, such as material, light, geometry, etc. This section covers the fundamentals as well as recent approaches to estimating material properties and geometry (surface normal) from a single image.

### II.3.1. Reflectance Estimation

As part of this research, we aim to estimate the material properties of a given object. In the following section, we will first review the fundamentals of modeling appearance properties (materials), and then we will discuss the state-of-the-art of finding the underlying components of an image, particularly the material properties.

#### II.3.1.1. Preliminaries

The reflection property of a material is described using bidirectional reflectance distribution function (BRDF) [NRH77]. The BRDF is a four-dimensional function that depends on the direction of the incoming light and the viewpoint. It can be measured in a variety of ways: One approach is to completely sample it using a device called a gonioreflectometer. Figure 25 shows a gonioreflectometer created by Murray-Coleman et al [MS90]. This device consists of a light source and a reflectance detector. The material is placed on the sample area and the BRDF is then measured by repositioning the material, as well as the light source and reflectance detector, in every possible way

(4 degree of freedom). This method produces large tables of data, which can then be approximated using functions.

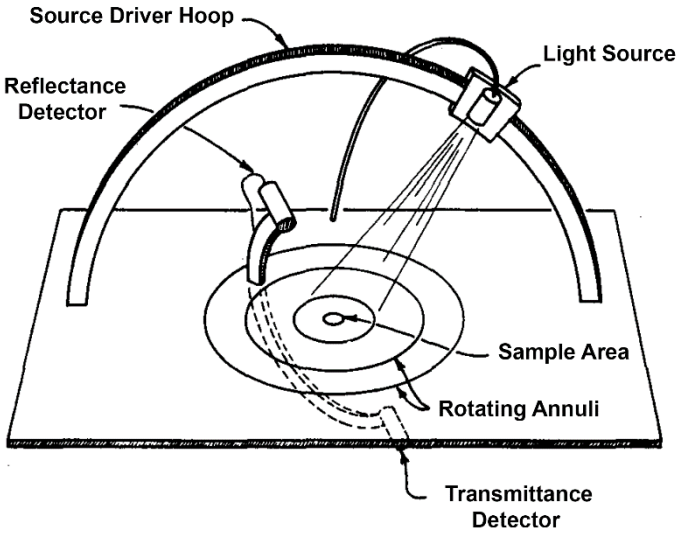


Figure 25 : The BRDF can be measured directly using a gonioreflectometer [MS90].

Another approach of estimating BRDF is approximating it using analytical models. There are several models available, including Lambertian, Phong, Ward, Cook-torrance, etc. Some of these models are only suitable for a certain type of materials.

The Lambertian BRDF is the simplest model and describes an ideal diffuse reflectance. In this model the BRDF is independent of the viewing angle, and the surface reflects the light evenly in all directions. However, there aren't many objects in real world that adhere to this model; Under focused light, even the rough surfaces exhibit some specular reflections.

The Phong model [P75] is a simple widely used model for simulating specular reflections. This model extends the Lambertian model by two more components. Thus, the BRDF is calculated by the sum of ambient, ideal diffuse (Lambertian) and specular components. But then again, since this model is empirical and not physically-based, it is possible that it reflects more light than it receives (No energy conservation). Therefore, the result may be unrealistic.

There are also physically-based models that describe the BRDF based on the microfacet theory [TS67]. They assume that every surface is composed of set of small mirrors (facets) and the roughness of a surface can be specified by altering the distribution of the orientation of these facets.

### II.3.1.2. Related work

The problem of determining the underlying properties of an image can be approached using two different ways: intrinsic image decomposition and inverse rendering.

**Inverse rendering:** The objective of inverse rendering is to represent a scene from an image in such a way that it can be rendered again from a new viewpoint. This is a hard task since we need to estimate lighting information, the shape of the objects (geometry) and their appearance (material properties) given only one image. In recent years, this problem is tackled using two approaches of neural rendering [TFT20] and differentiable renders [ZJL20]. Although neural rendering methods generate high-quality and realistic results when re-rendering from a novel viewpoint or relighting a scene under new lighting parameters, their model representation is implicit, and thus the manipulation of the scene parameters is somewhat constrained. Differentiable renders are another active research topic that appears to be very promising, but it still requires extensive research due to its difficulty.

**Intrinsic image decomposition:** This approach considers that each image is composed of two image layers: one that exhibits the effects formed by scene illumination (such as shadows) and one that exhibits material reflectance (color and texture). Therefore, to obtain the underlying properties of an image, it is sufficient to divide each image into these two layers; that is, each pixel is formed of the multiplication of two RGB values. The majority of approaches in this category assumes a Lambertian material. In the past, this task was accomplished by finding local features in the images such as edges and its primary application was to edit images [BKPB17] (removing shadows, re-coloring the objects, etc). Nowadays, deep learning is used since it can learn more than just the local features and incorporates the global semantics of each image [GRCL22].

In the next section, we will review the studies that concentrate solely on acquiring reflectance properties of an object from a single image particularly when using a deep learning approach:

Georgoulis et al. [GRR18] proposed a two-step learning-based approach to estimate the BRDF and the illumination from a single color image. They estimated a reflectance map [HS79] from a given color image and the corresponding binary mask. In the first step, then divided it into the BRDF and the illumination in the second step. Encoder-decoder architecture is used for both two steps. The output illumination is represented as an HRI spherical illumination map, and the BRDF is represented using the Phong model. Georgoulis et al. also proposed different approaches for each of the two steps; for example, they proposed a direct and an indirect approach for the first step of

reflectance map estimation. The direct approach consists of directly estimating the reflectance map using neural networks and the indirect one consists of first estimating the object's normal map using neural networks, then reconstructing a sparse reflectance map from the normals and eventually interpolate sparce reflectance using another network architecture. The complete approach is illustrated in Figure 26.

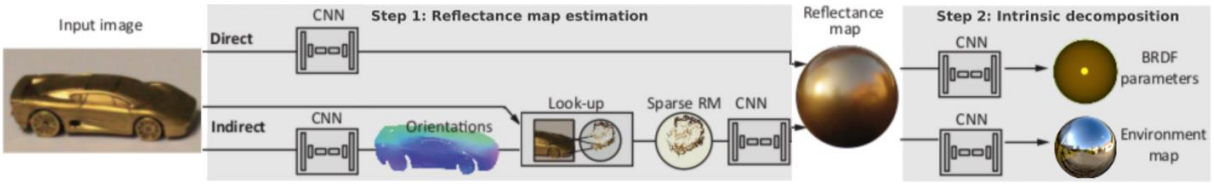


Figure 26 : Georgoulis et al. [GRR18] proposed a two-step learning-based approach to estimate the BRDF and the illumination from a single color image

Georgoulis et al made few assumptions in their approach including that there are no shadows in the image, and the object is chosen from a known class (such as cars) and has a single material. Also, they performed the image segmentations manually.

Later, Meka et al. [MMZ18] proposed an approach using deep learning that estimates the BRDF of a material based on Blinn-Phong reflection model. Their proposed system consists of five sub-networks and is motivated by the process of physical image formation. First, the *SegmentationNet* takes a color image as input and estimates a binary mask. Next the binary masked is applied to the image and the masked image is fed into *SpecularNet*, which estimates a specular shading image. This specular shading image represents the normalized specular reflections of the object. It is then fed into the *MirrorNet,* which estimates an image demonstrating the high-frequency illumination arriving at the surface of the object. In other words, this network removes the surface roughness. All three networks mentioned previously have an encoder-decoder architecture (U-Net). Eventually, the two networks *AlbedoNet* and *ExponentNet* estimate the material parameters using the results of the previous networks. Figure 27 illustrates their pipeline. The *AlbedoNet* is defined as a regression problem, whereas the *ExponentNet* is defined as a classification problem.

Meka et al. also generated a synthetic dataset for training all the networks. Their images were rendered with an object in the center in an indoor setting.

The proposed system is real-time and performs well on general shapes. However, Meka et al. made a few assumptions in their work, such as the object not casting a shadow on itself (no self-shadowing) and not emitting light.

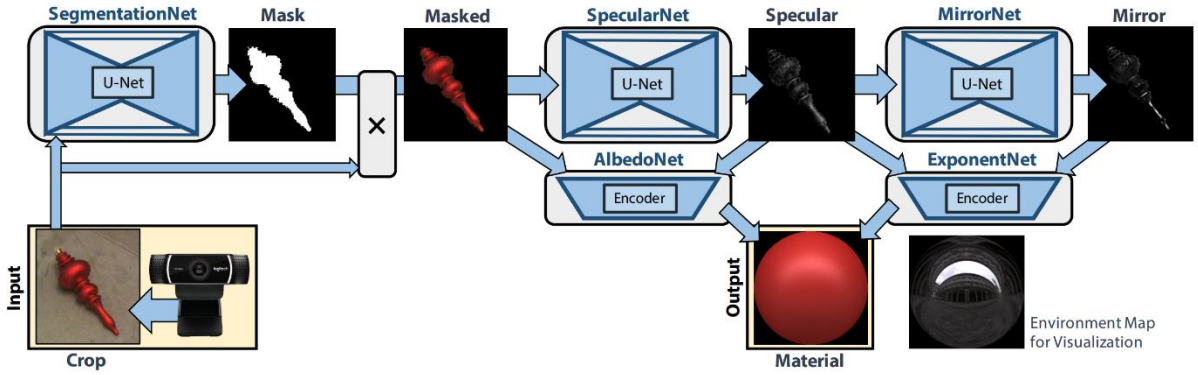


Figure 27 : Meka et al. [MMZ18] proposed a pipeline consisting of five sub-networks that is motivated by the process of physical image formation.

## II.3.2. Surface Normal Estimation

Research and technology have advanced significantly in recent years, resulting in the rise of new applications that demand 3D geometric understanding. A few examples of these applications include robot navigation, robot manipulation, augmented reality, etc. The 3D geometric information can be represented in numerous ways including pixel-maps (depth map, normal map), point-cloud, mesh, implicit surfaces.

One of the common types of geometric information is normal map which is a pixel-map that describes the 3D surface orientation at each pixel. The X, Y and Z of the normal vector are usually stored in R, G and B of color image. In this section we go over the state-of-the-art in normal map estimation from a single image.

Wang et al. [WFG15] proposed a pipeline to estimate the normal map from a single image on two scales: global and local. On the global scale, they trained a network to take a color image as input and estimate a global normal map and a room layout (under the Manhattan world assumption) as output; on the local scale, they trained a network that uses sliding window on the color image and feeds the local patches as input to the network. The network estimates local normals and local edges as outputs. The local edges are classified as concave, convex, or occlusion. Although both the normal estimation and room layout estimation are on continuous spaces and are regression problems, Wang et al. reformulated them as classification problems. Specifically in normal estimation, they used the triangular coding method proposed by Ladicky et al [LZP14] to reduce regression as classification. Eventually, they trained a fusion network to integrate the previous results and generate a more accurate normal estimation.

Eigen et Fergus [EF15] designed a single multiscale convolutional network to regress depth map, normal map, and semantic labels. Their model architecture is composed of three scales; the first scale takes the color image as input and predicts a global estimation as output; the later scales then refine the output to generate a high-quality result.

Zhang et al [ZS17] provided a synthetic dataset of various interior environments and analyzed the influence of different types of rendering on neural network training. They used a modified U-Net [RFB15] structure with VGG-16 encoder for normal estimation and achieved the state-of-the-art results.

## II.4. Depth Completion

Low-cost RGB-D cameras are often incapable of accurately measuring the depth of objects near their boundaries or the surfaces that are transparent, shiny, thin, too close, or too far away from the camera. These poor measurements may appear as incorrect or missing values (holes) in the depth map.

The fast marching method (FMM) was first proposed by Telea [T04] to inpaint color images. Later Liu et al [LGL12] extended the FMM for depth inpainting. In their method, they choose the inpainting order based on color similarity and distance to the hole boundary. Therefore, the pixels on the hole boundaries surrounded by neighbors with similar colors are inpainted first. The value for the missing depth is calculated by using a weighted average of the neighbour depth pixels. Eventually, they used an edge-preserving filter on the completed depth to reduce the noise. However, since the incorrect regions were not removed prior to depth propagation, the result depth map may contain incorrect values. Huang et al [HHC14] proposed a method to locate and remove the unreliable depth values before depth completion. They found all the edges on the color image using Canny edge detection, then enlarged them to obtain the unreliable regions in the depth map. They then checked the reliability of each pixel in the unreliable region by comparing its depth to its neighbors. After removing the incorrect depth pixels, they used the fast marching method [LGL12] to find the order of inpainting and completed the depth using a joint bilateral filter.

In recent years, the latest advances in artificial intelligence and deep learning have led to a wide range of data-driven approaches. Zhang et Funkhouser [ZF18] proposed a method using convolutional neural networks and optimization to complete the missing depth in large indoor scenes. Through their work, they explained that depth completion has some specific challenges. These challenges are described below:

1. Training data: The RGB-D images captured by low-cost cameras lack ground-truth data, where the holes are filled. Many depth estimation methods only train on the observed depths and can in the best case, reproduce them. These methods can't fill the holes because these areas have different properties. Therefore, Zhang et Funkhouser used Matterport3D [CDF17] dataset to create a large-scale training set of 105,432 RGB-D images. They used multi-view reconstruction and rendering to fill the missing values of the depth map and obtain ground-truth data. They prepared this data from 72 real-world scenes. Their reconstructions reduce missing pixels by 64.6 percent and also reduce the noise in ground-truth depth by averaging between the depth of different viewpoints.

2. The Method: Zhang et Funkhouser conducted experiments to determine the best type of input and output of the neural network for depth completion, as well as some experiments on the ground-truth data and loss calculation, which we discuss in detail below.

- The best type of output: They first tested whether it is better to predict completed depth, depth derivatives [CS, S16], or local differential properties of depth (normal map and boundary map). According to their experiments, predicting the normal map and boundary map is much easier and more accurate. Therefore, they trained convolutional neural network to generate boundary map and normal map from a single color image.
- The best type of input: They tested various types of input to determine which produced the best results in a normal estimation task; a single color image, a depth map, or a color image with the corresponding depth (RGB-D). Their tests demonstrate that using only color image produces better results.
- The ground-truth data: They experimented which ground-truth data is better for estimating normal map; using normals of the rendered depth map or computing the loss only on the normals of available pixels in raw depth map. Their experiments show that using rendered depth improves the normal estimation result. This is because the rendered depth contains more information (depth of the holes) and less noise than the raw depth. Furthermore, while using rendered depth as ground-truth, they tested whether training only on pixels inside holes or training on all pixels yielded better results. This question arises because the pixels in holes have different color characteristics than other pixels (for example, they are too far away or occur due to specular reflections), so training only on them may produce better result, but this also reduces the amount of training data. After testing, they came to a conclusion that training on all pixels produces better results.

Next, they used an optimization approach with the following objective function to complete the depth:

$$E = \lambda_D E_D + \lambda_S E_S + \lambda_N E_N B$$

$$E_D = \sum_{p \in T} \| D(p) - D_0(p) \|^2$$

$$E_N = \sum_{p,q \in N} \| < v(p,q), N(p) > \|^2$$

$$E_S = \sum_{p,q \in N} \| D(p) - D(q) \|^2$$

where $E_D$ is the distance between the estimated depth and the raw depth at pixel p. $E_N$ uses the dot product to assess the consistency between the estimated depth and the predicted surface normal. $E_S$ encourages neighbour pixels to share similar depth values. B has a value between [0,1] and down-weights the $E_N$ based on the predicted probability a pixel is on the boundary.

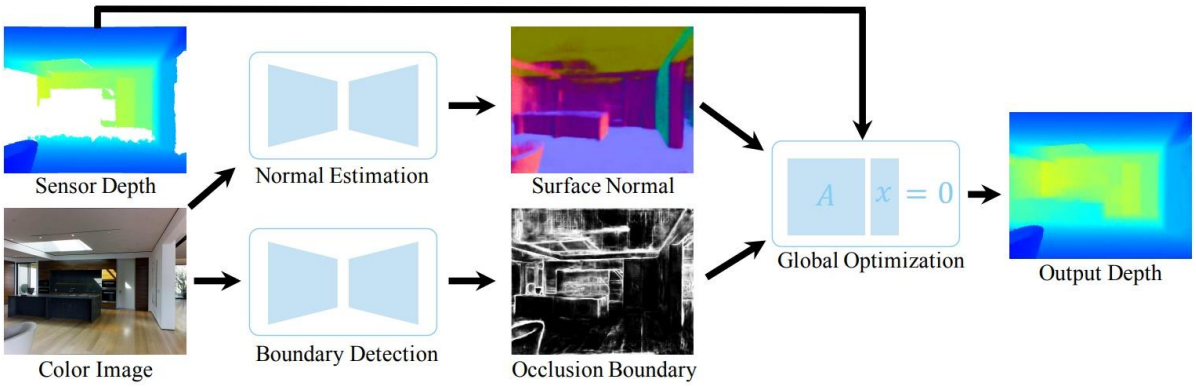


Figure 28: Proposed pipeline of Zhang et Funkhouser [ZF18]; They estimate normal map and boundary map from a color image and use them to complete a depth map using a global optimization method.

Sajjan et al [SMPN20] modified the previous method and proposed the “ClearGrasp” algorithm to correct the depth of transparent objects for robotic manipulation. Transparent objects exhibit both specular reflection and refraction and low-cost RGB-D cameras often confuse their depth with the depth of the surfaces behind them. Sajjan et al trained neural network to estimate normal map, boundary map and a mask of

transparent objects from a single color image. They used the mask to locate the transparent objects and remove their depth completely. The boundary estimation network estimates three classes (labels) for each pixel: non-boundary pixel, contact pixel (where the objects are in contact with other surfaces/objects) and discontinuity pixel (where depth gets discontinued; meaning that the depth of the object and its surroundings are different). To train the network, they created a large synthetic dataset of 50,000 RGB-D images. They prepared the data using Synthesis AI's platform and rendering with Blender Cycles. They used 5 types of objects for training and 4 types for testing. They used physics to produce a random scene by releasing objects onto a surface or a box. To test the network, in addition of the synthetic test set, they prepared 286 real-world RGB-D dataset.

At last, they used the optimization approach proposed by Zhang et Funkhouser [ZF18] to complete the depth. They used the same values for $\lambda$ as Zhang et Funkhouser in their experiments.

They demonstrate the effectiveness of their pipeline on a robotic arm using a state-of-the-art grasping algorithm to grasp transparent objects and show that it significantly improves the results. However, their method still requires further improvement; they need to improve the pipeline to reduce the error under varying illumination conditions as well as where there are sharp shadows and caustics. In addition, the cluttered environments make their boundary estimation more prone to error.

# Chapter III. Depth Completion for Close-Range Specular Objects

## III.1. Introduction and Overview

The common low-cost RGB-D cameras suffer from some limitations in depth sensing. We used Intel RealSense D435 to observe some of these limitations. This RGB-D camera is low-cost and lightweight, and it employs active stereo technology. The following are our observations based on the captured data:

1. The depth map contains missing depth values, mainly near object boundaries and inside specular objects. Figure 29 illustrates a few examples of this problem. In this figure, the first column shows the color image while the second column shows the colorized depth map. As we can see, the depth values at the object's boundary are usually missing (shown as black areas in the colorized depth map). The missing area in the third row is significant due to the object's specular reflection.
2. There are pixels with incorrect depth values inside the specular or transparent objects. Figure 30, shows an example of this issue. The first row shows the color image and the raw colorized depth map while the second row shows two normal maps; one estimated from the color image and the other calculated from the depth map. Here, the normal map estimated from the color image serves as the ground-truth. We can observe that the depth of the bottom part of the box is completely incorrect, as its normals have the orientation of the table.

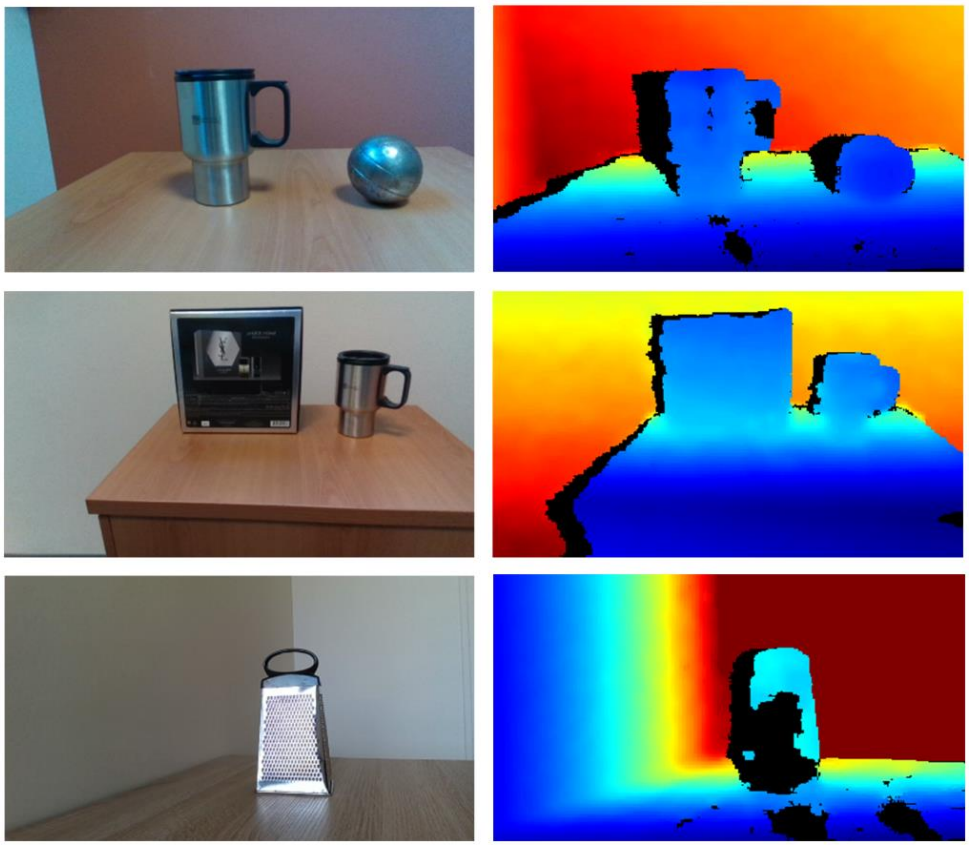

Figure 29: The Intel RealSense D435's depth map contains missing values, which mainly occur near object boundaries and inside specular objects. Here, the first column shows the color image while the second column shows the colorized depth map. The missing depth is shown in black on the depth map.

3. There are some incorrect depth values near the object's boundaries, which indicates that the camera confuses the object's depth with the background near borders. This problem also occurs between two nearby objects.
4. The depth contains a lot of noise, which is particularly visible on the curvature of specular objects.

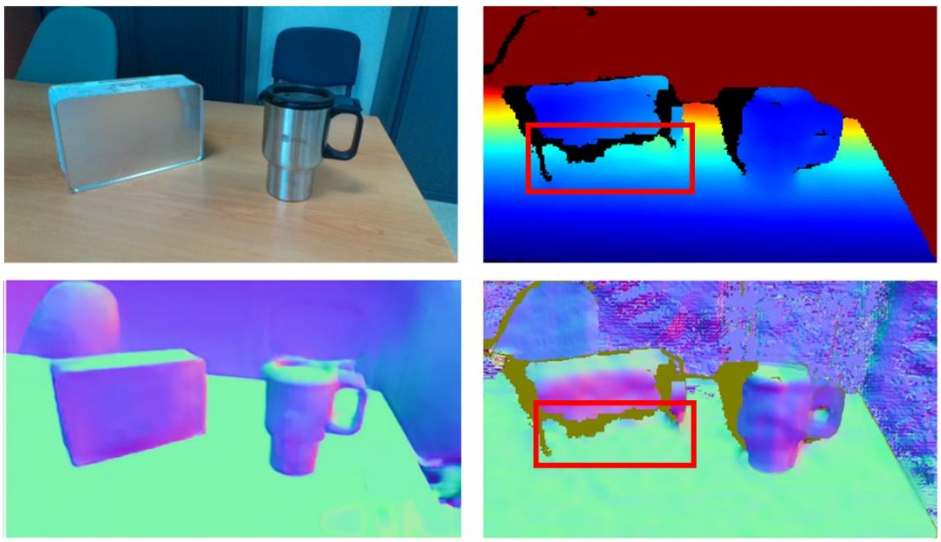

Figure 30 : The first row shows the color image and the raw colorized depth map while the second row shows two normal maps; one estimated from the color image and the other calculated from the depth map. The normal map estimated from color image is used as ground-truth. We can observe that the depth of the bottom part of the box is completely incorrect since its normal map has the orientation of the table.

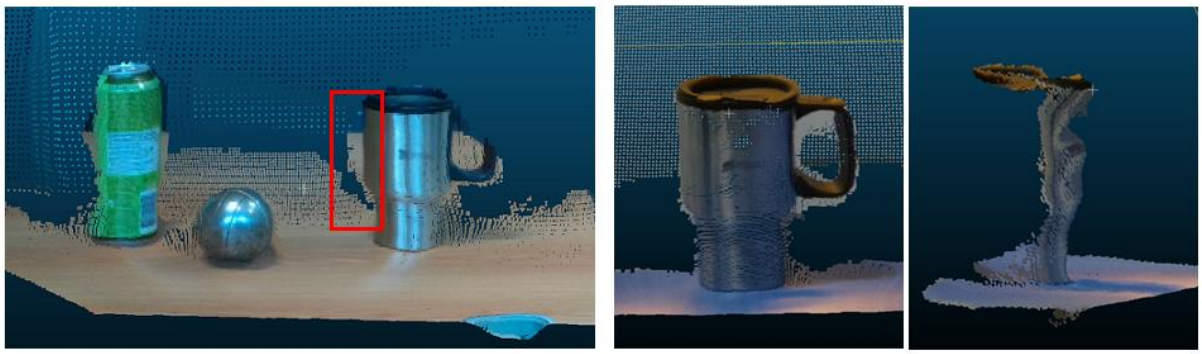

Figure 31 : The camera confuses the depth between the object and the background near the borders (Left). The depth also contains noise which can be seen on the noisy curvature of a mug (Right).

In this work, we present an approach to correct and complete the depth map of close-range specular objects. Our approach consists of several parts:

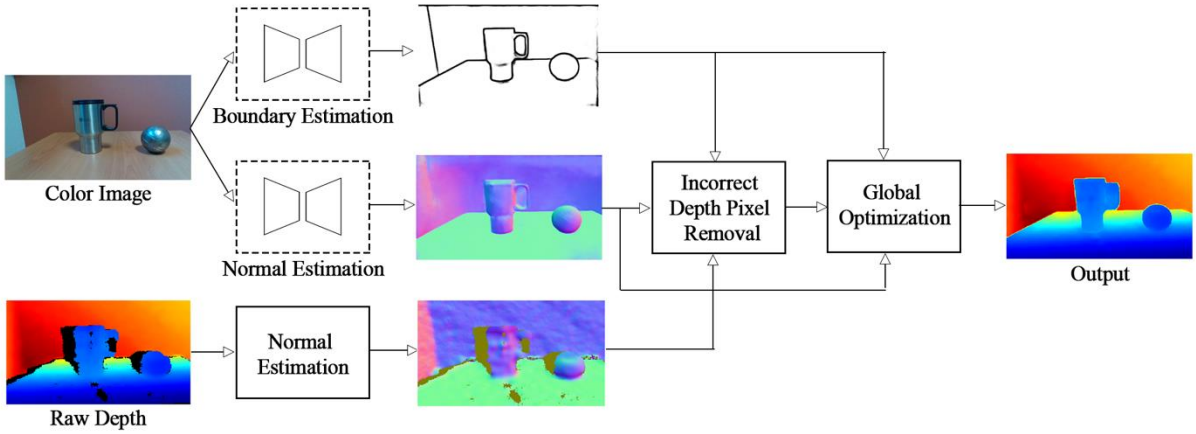


Figure 32 : Overall illustration of the proposed pipeline. First the networks estimate the boundary and normal map from a single color image. Next the incorrect depth is located and removed using the output from the networks. Eventually, the depth is filled out using a global optimization approach.

We first create a random-scene generator to generate synthetic images for training neural networks. In the next step, we train a boundary estimation and a normal map estimation network on our training data, then use their outputs to locate and remove invalid areas of the depth map. These invalid pixels are detected and removed in three steps; first the object's boundaries are removed. Second the regions where there is a significant difference between the normal map estimated from color image and the one estimated from depth map are removed. Third, a morphological transformation is used to eliminate the remaining noise. At last, we complete the depth using an optimization approach. The proposed pipeline is illustrated in Figure 32.

## III.2. Dataset Generation

We used Blender to generate our own dataset to train neural networks for normal estimation and boundary detection. We started by creating a plane with a random texture to use as the underlying surface for placing objects on. The texture of the plane is chosen at random from a set of 82 textures [URL3]. Following that, we placed a camera in a random location around the plane, with the constraint that it always focuses on the plane.

Next, we set up the environmental lighting by randomly choosing an HDRI file (High Dynamic Range Imaging) from a collection of 120 indoor environments [URL4]. We then rotated the HDRI at random to add more randomness to the rendered results.

Following that, we carefully selected a set of 9 Blender primitive shapes that varied in their geometric properties, such as whether they had a smooth surface or sharp edges or holes or not. The selected shapes are illustrated in Figure 33. Then we randomly selected a number of objects from the set and resized them to a random size. Afterward, we applied a random texture to each object from a collection of 47 textures (including the Blender's checker pattern or a simple color) [URL3]. We also randomly selected a roughness value to give the objects a rough or specular appearance.

Next, similar to the approach of Sajjan et al [SMPN20], we applied the Blender physics system to the objects, and dropped them on the plane. This would create a random scene, as the objects collide with the plane and one another before coming to rest in a random location.

Figure 33 : we carefully select a set of 9 Blender primitive shapes with varying geometric properties to generate a random scene.

We used GPU to generate three types of data for each scene: a rendered color image, a corresponding normal map and a boundary map. To produce high-quality rendering images, we used Blender Cycles, which uses a physically-based path-tracer. We set the rendering resolution to 256x256 pixels.

The normal map was saved in OpenEXR format, while the color image and boundary map were both saved in PNG formats.

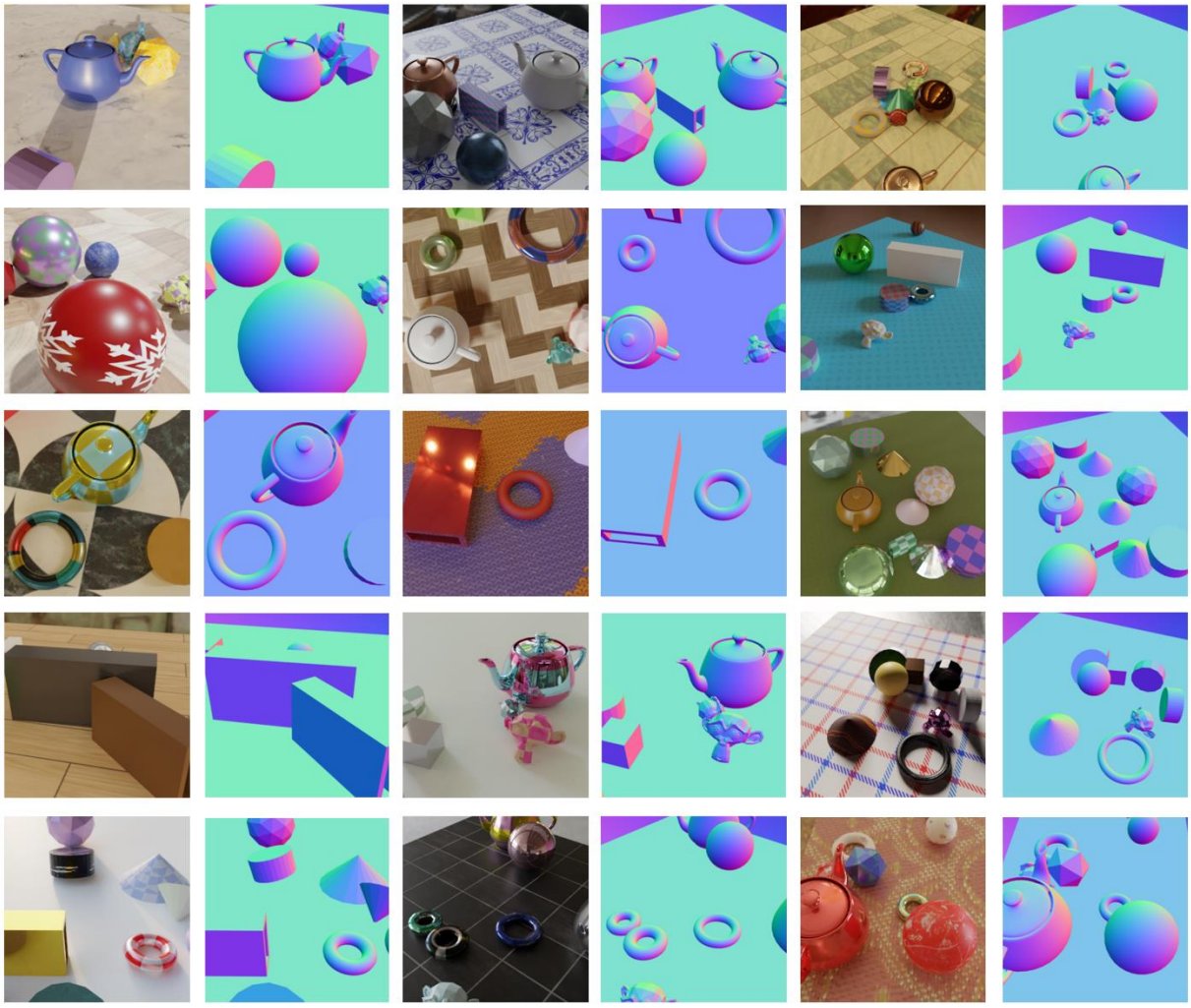

Figure 34: Dataset generated by our scene generator to be used for training the normal estimation network

When generating the normal map data, there is a case where the HDRI is visible through the camera view. In order to prevent the normals of those locations from remaining empty, we create a bounding-sphere around the plane and the objects. This does not pose a problem while training the normal estimation network since the background information is irrelevant in our scenario. Next, we change the orientation of the normals from world-space to camera-space and normalize them between -1 and 1 values.

We use this scene generator program to generate a large amount of data for the following steps. Figure 34 and Figure 35 illustrate the dataset used for training normal estimation and boundary detection networks.

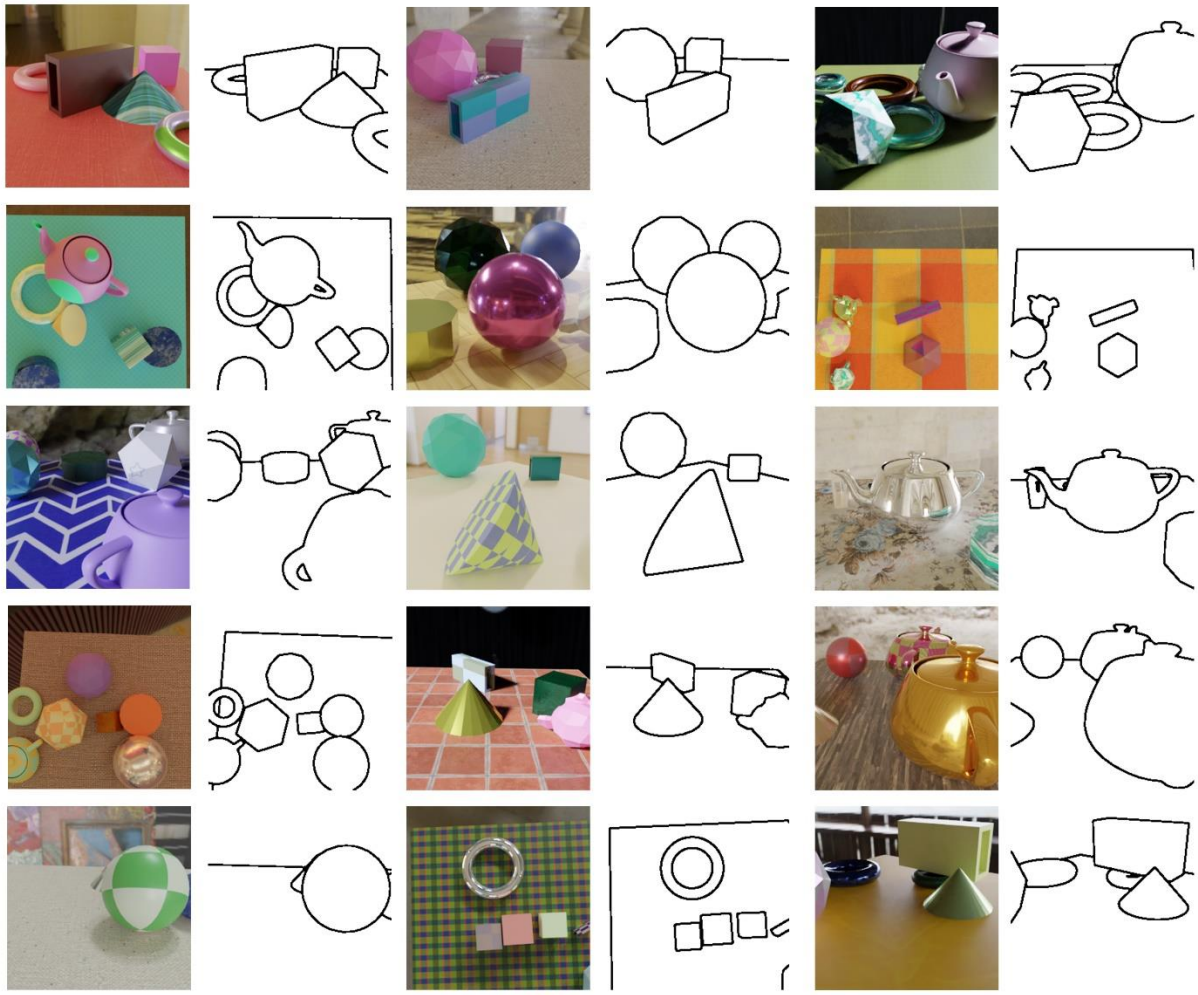

Figure 35: Dataset generated by our scene generator to be used for training the boundary detection network

## III.3. Network Architecture and Training

We used an encoder-decoder architecture for both normal estimation and boundary detection tasks. However, the choice of which encoder-decoder architecture to use was not evident. Therefore, we compared different models to choose the best one for our data.

### III.3.1. Normal Estimation

We performed a comparison between U-Net [RFB15] and DeepLabV3 [CPSA17] encoder-decoder architectures to find the best model for our data. The DeeplabV3 is trained with the Resnet101 [HZRS16] backbone while the U-Net is trained with various backbones of Resnet101, Inception-V4 [SIVA16] and VGG-16 [SZ14]. All the backbones (encoders) that we used were pre-trained on ImageNet dataset [DDSLLF09] [P19].

We modified the output layer of the models to produce a three-channel output. Therefore, each output channel represents one of the three components of the normal vector x, y and z. We then normalized the three-channel output to the L2-norm.

We used the data that we generated using our scene generator to train the networks. We performed few pre-processing on the input data. First, we augmented them using Gaussian blur. Second, since all the encoders were pre-trained on ImageNet, we normalized the input data the same way of pre-training for faster convergence. The normalization is performed this way for each channel c of the data:

$$output[c] = (data[c] - mean[c]) / standard\ deviation\ [c]$$

where the mean tensor is [0.485, 0.456, 0.406] and standard deviation tensor is [0.229, 0.224, 0.225]. These values were computed over millions of images of ImageNet dataset. This is especially good practice if the training datasets contain “common” natural images similar to ImageNet (such as indoor and outdoor scenes, animals, people, etc). For other types of “un-common” images (such as images taken under unusual lighting or medical or satellite images) it is better to calculate the mean and standard deviation directly from the training dataset.

We calculated the loss function based on the element-wise dot product of the output and the synthetic ground-truth data. We also shuffled the training data at each epoch during training to prevent the network from learning the order of the data.

We used the Adam optimizer with learning rate of 0.0001 but reduced it by the factor of 0.1 when there were no improvements after more than two epochs.

We trained the networks on 10,000 synthetic data that we generated and evaluated each network based on the common metrics used in normal estimation. Table 1 shows the results.

Table 1: This table shows the comparison between the networks on the validation set based on the common metrics used in normal estimation evaluations.

| Model | Backbone | Mean | Median | 11.25° | 22.5° | 30° |
|---|---|---|---|---|---|---|
| DeeplabV3 | ResNet101 | 16 | 6.4 | 69.4 | 87.4 | 92.0 |
| U-Net | ResNet101 | 16.21 | 9.2 | 60.1 | 87.5 | 93.4 |
| U-Net | Inception-v4 | 15.61 | 8.0 | 67.1 | 89.1 | 93.2 |
| U-Net | VGG-16 | 16.49 | 8.3 | 62.9 | 86.5 | 92.5 |

We also tested the networks on some real-world data during the training. The results from DeeplabV3 were always blurrier than those of U-Net. However, changing the encoders (backbone) of U-Net architecture had little effect on the output. Figure 36 shows an example.

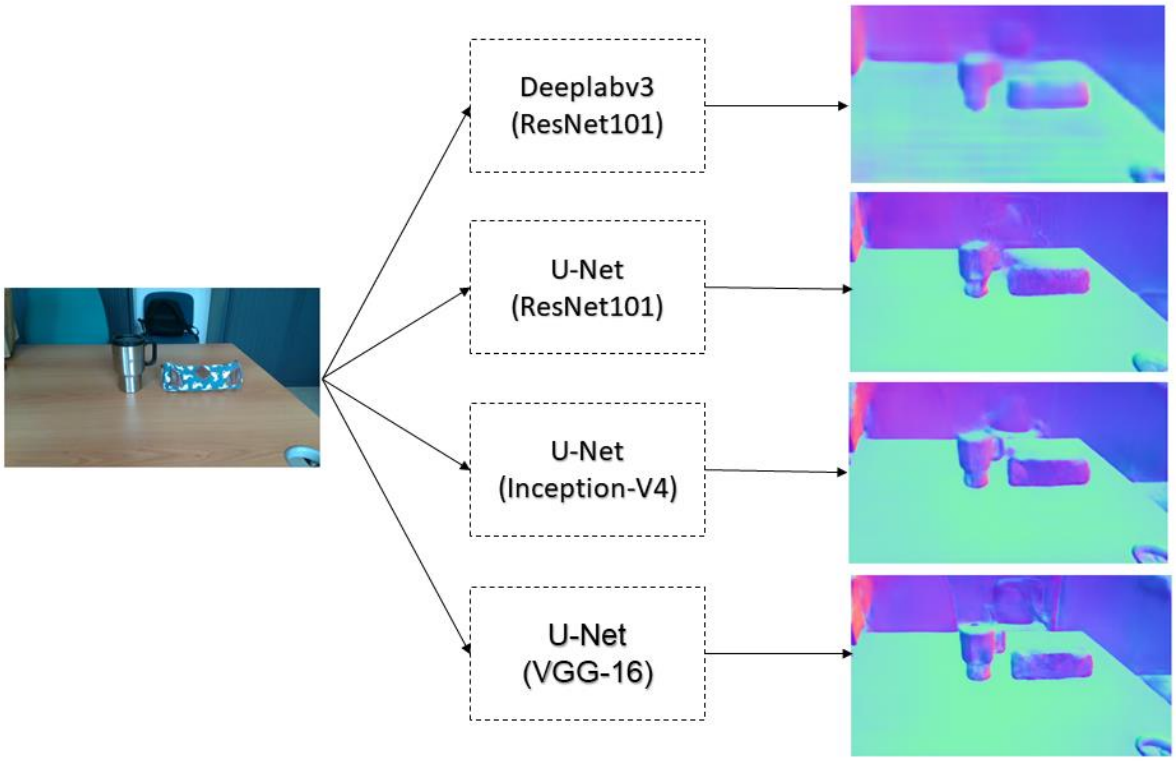


Figure 36: We tested the networks on some real-world data during the training. The results from DeeplabV3 were always blurrier than those of U-Net. However, changing the encoders (backbone) of U-Net architecture had little effect on the output.

Table 2 : Evaluation metrics of our normal estimation model for the synthetic test set

| | Mean | Median | 11.25 | 22.5 | 30 |
|---|---|---|---|---|---|
| Our model | 24.2 | 18.3 | 38.7 | 54.3 | 99.8 |

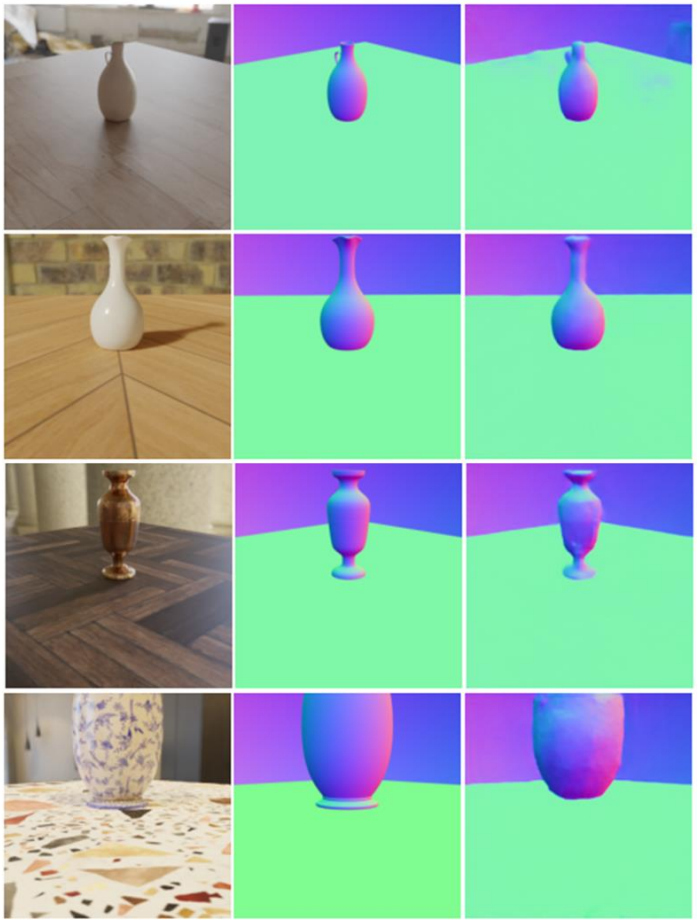

Figure 37: The results of the network on some new shapes with new textures. The left column shows the color image, the middle one shows the ground truth normal map and the right one shows the result from our network.

We decided to use the U-Net architecture with Inception-v4 encoder as our model. After choosing the model, we reduced the learning rate and re-trained it on additional 5000 synthetic images of very close-range objects. We also used the previously described data pre-processing (augmenting the data using the gaussian blur and

normalizing them). The reason for training in this way was that in the first step, the network mostly learns the shape and the angle of the table relative to the camera viewpoint as well as an approximate shape of the objects on the table. In the second step, it will learn the shape of the objects more precisely and refine its predictions. By using a low learning rate in this step, we prevent the network from forgetting the previous step. We evaluated the performance of the network on 100 previously unseen synthetic images generated by our scene generator.

Table 2 shows the results of the evaluation of the final version of our model on the unseen test set. In addition, Figure 37 shows the network's results on some synthetic data containing new shapes and new textures [URL6].

### III.3.2. Boundary Detection

We chose U-Net architecture with Inception-v4 encoder as our boundary detection network similar to the normal estimation task. In the training data that we generated, each pixel on the boundary has a value of zero, while all other pixels have a value of one. Therefore, we changed the output layer of the model to generate one-channel output to perform a per-pixel binary classification. We applied the sigmoid activation function to the output layer which assigns each pixel a value between zero and one. This per-pixel value p can be interpreted as the confidence that the pixel belongs to the first class, while 1-p can show the confidence of that pixel belonging to the second class. We calculated the loss by using the binary cross-entropy function. As the number of pixels who belong to the non-boundary class is much higher than the ones belong to boundary class, we used a loss weight that is ten times higher for boundary pixels than for non-boundary pixels. This is suggested by Yang et al. [YPCLY16]. We used 5000 images to train the network which we augmented with gaussian blur and changes to their contrast, brightness, saturation, and hue.

## III.4. Incorrect Depth Pixel Removal

Before completing the depth, it is critical to remove the incorrect depth to prevent it from propagating. We perform this in three steps:

1. As stated in the introduction to this section, the RGB-D cameras usually confuse the depth near the boundary of the objects; the confusion can occur between the adjacent objects or between a foreground object and the background. Thus, we start this section by removing the depth of the boundaries. For this task, we use the boundary map estimated by the neural network described in the

previous section. Since the boundary map is a grey scale image, we first convert it to binary; which means that it would only contains black or white values. This can be achieved simply by using the mean of the pixels intensities as the threshold, but we preferred to use the Otsu's method which determines the threshold of the greyscale image automatically. This method is a well-known image processing technique that divides the pixels into two classes using the image's histogram. After creating our binary mask, we apply it to the depth map to remove the boundary depth.

2. Next, we compare the normals of the depth map to the normals estimated from color image using the dot product. We perform this step to remove the incorrect areas on the depth map that appear due to the specular reflections. We remove the depth where the difference between the angle of corresponding normals is more than 30 degrees. We choose this value experimentally because the error of normal map estimation network is almost always less than 30 degrees relative to the ground truth value. Additionally, we don't want to remove the depth excessively, and if an application demands for a depth map with higher quality and precision, the depth of the areas with small amounts of normal differences can get rectified after depth completion using filtering.
3. Finally, we use morphological transformation to remove any remaining small noise from the previous two steps. Firs we create a binary mask from the depth map, with a value of zero where there is a hole and a value of one where the depth is present. Then we use morphological opening with circular structure kernel of the size 5x5 on the mask to remove the noise. Then we apply the mask to the depth map.

## III.5. Global Optimization

In this step, we complete the depth using the optimization algorithm proposed by Zhang et Funkhouser [ZF18]. Chapter 2, section 4 provides a thorough explanation of the optimization algorithm. The approach takes the normal map and boundary map estimated from color image as input and uses them as a guide in the depth completion process.

## III.6. Results and Discussion

We tested the proposed approach on real-world data. The acquisition process is as follows:

We set up a test environment in an indoor scene consisting various objects on a table. Some of the objects are similar to the synthetic primitive shapes we used to train the network, while others have new shapes that the networks have never seen before. The scene is set up under natural lightning.

To capture the depth and the color image, we used an Intel real-sense RGB-D camera. We set the depth and color stream resolutions to 424x240 pixels. In order to give the auto exposure time to adjust, we started streaming with the camera but skipped the first 20 frames. Then, using the Intel RealSense SDK [URL5], we aligned the color frame and the depth frame since they were obtained using different lenses. To reduce the noise in the depth map, we used a temporal filtering, which is a post-processing filter available in the SDK. The temporal filter modifies the depth values based on the previous frames. It can add depth values where the depth is missing or find the incorrect depth and change its value based on the history values that it keeps. This filter is only suitable for use in static scenes.

We then pass the color images through the normal estimation and boundary detection network. The output of each network is illustrated in Figure 38. The first column shows the input image, the second shows the output of boundary detection network, and the third shows the normal map estimated by normal estimation network.

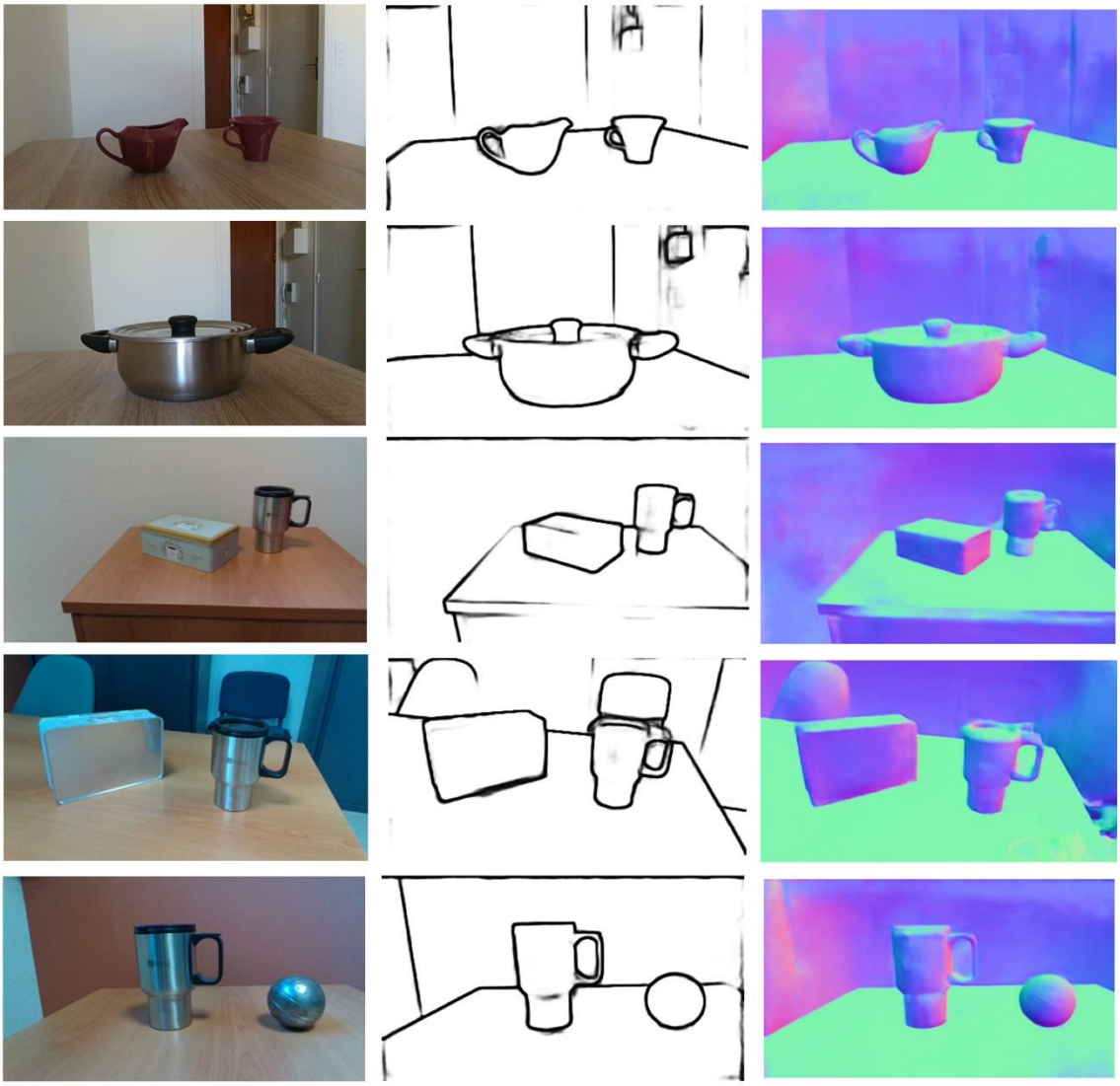

Figure 38 : The first column shows the input image, the second shows the output of boundary detection network, and the third shows the normal map estimated by normal estimation network.

We then continue through the pipeline and perform the incorrect depth removal and the depth completion steps. We perform all the training and the testing steps on a desktop computer equipped with an Intel Core(TM) i9 2.80GHz CPU with 16 GB RAM and NVIDIA GeForce RTX 2080 GPU. The entire process of our proposed approach takes about 2 seconds for given data with a size of 424x240; this includes 0.3 seconds for normal and boundary estimation from color image and incorrect depth removal from the depth map, as well as 1.8 seconds for the optimization-based depth completion step.

We compared the result from our proposed method to the results from the work of Zhang et Funkhouser [ZF18]. We downloaded the code they provided from their GitHub page and used it without modifying it. In the depth completion step, we used the same values as Zhang et Funkhouser [ZF18] which are: $\lambda_D = 1000$, $\lambda_N = 1$ and $\lambda_S = 0.001$. Figure 39 illustrates the comparison; the first column shows the color image, the second shows the raw depth map captured by the Intel-realsense camera, the third shows the results of the work of Zhang et Funkhouser and the last column shows the result of our proposed approach. As we can see, our proposed approach significantly improves the depth map quality of close-range objects. Take note of how the object boundaries have become sharper and the depth of the boxes in the rows three and four have become more consistent and accurate.

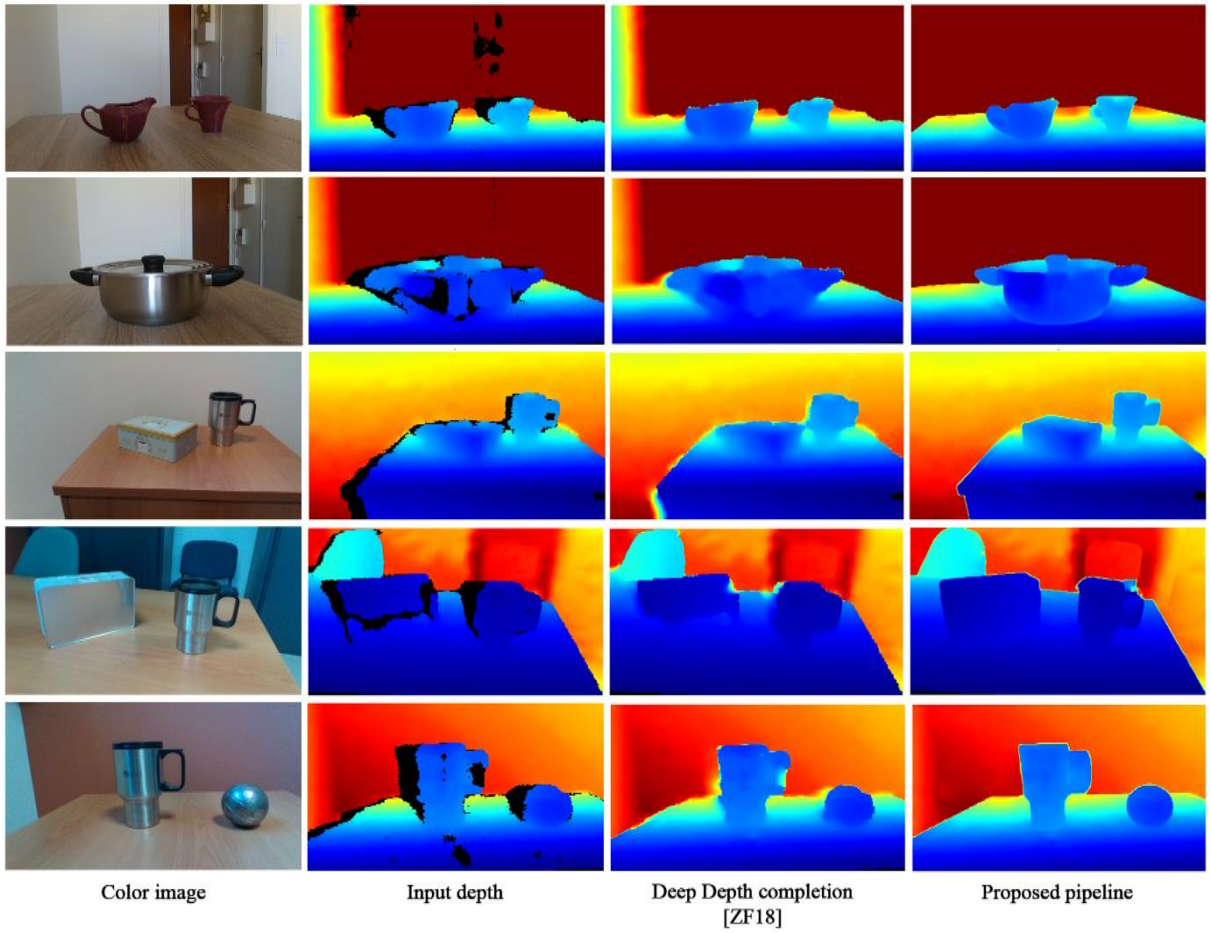


Figure 39: The first column shows the color image, the second shows the raw depth map captured by the Intel-Realsense camera, the third shows the results of the work of Zhang et Funkhouser [ZF18] and the last column shows the result of our proposed approach. In the first and second rows, the depth further than one meter is clipped to improve the visualization of the colorized depth map.

Nevertheless, there is a disadvantage to our approach that may occur, but is very uncommon; it occurs when the depth of a region is incorrect, but their normals are correctly oriented. As a result, our approach won't be able to correctly locate the incorrect region of the depth map since it depends on the normals. Figure 40 shows an illustration of this problem; since the incorrect region could not be correctly located, the inaccurate depth values are propagated during the depth completion step.

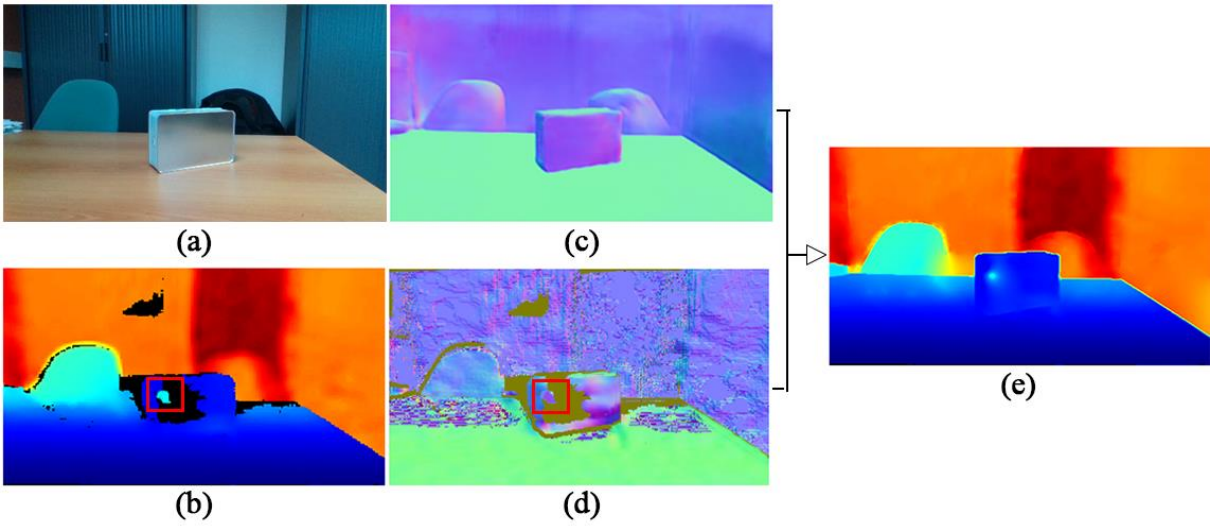


Figure 40 : illustration of a rare drawback in our approach; (a) color image (b) raw depth map (c) normal map estimated from the color image. (d) normal map of the depth map (e) the completed depth map. The highlighted area shown in the depth map (b) is where the depth values are incorrect but the orientation of their normals are correct. As a result, our approach won't be able to correctly locate this incorrect region and remove it.

## III.7. Conclusion

In this chapter, we proposed an approach to correct and complete the depth map of close-range specular objects. We made two contributions: first we developed a scene generator to generate large-scale synthetic datasets for training neural networks. In this way we would have pixel-perfect data for training networks. Second, we proposed a method to locate the incorrect areas in the depth map and remove them. We then completed the depth with the method proposed by Zhang et Funkhouser [ZF18]. Eventually, we tested our approach on the real-world data captured by the Intel-real

sense camera and compared our results to those of Zhang et Funkhouser. Our experiments show that our method greatly enhances the depth maps to use for close-range tasks. However, there are cases that may rarely occur and would prevent our approach from working effectively. These issues will be addressed in future works.

# Chapter IV. Reflectance Estimation

## IV.1. Introduction and Overview

Reflectance estimation is an important task in computer vision that is used in many domains including virtual reality and augmented reality.

In chapter 2 we recalled the fundamentals of reflectance acquisition, as well as the most recent methods that, like us, have tried to tackle the problem with neural networks. However, it is clear that the field of reflectance acquisition is extremely broad and we suggest the reader refer to the state-of-the-art [GGGDG16].

The problem of reflectance acquisition is particularly difficult due to the complexity of the materials. Thus, while a BRDF aims at representing the reflectance characteristics of homogeneous materials, a Spatially Varying BRDF (SVBRDF) can also be considered as a spatial collection of BRDFs distributed on the surface. There are very recent methods of acquiring SVBRDF as we try to consider on our side on neural networks, such as [DADDB19]. These types of methods, as we can see, impose many constraints in exchange for accurately solving this complex problem. These constrictions apply to:

- the lighting conditions
- the shape of the samples (often flat samples)
- the need for multiple images

We will not study these methods further faced with these constraints; instead, we decide to adopt the opposite reasoning and set the fewest constraints possible in the context of our cameras. As a result, we accept:

- variable and unknown lighting conditions (power, position)
- the arbitrary shape of objects
- a single input image (taken with a low-cost camera)

In this scope, there are a few methods that try to estimate the material properties of an object from a single image [MMZ18][GRR18]. However, their approaches consist of training multiple networks and still imposing numerous constraints on the object and the scene. Our objective is to define an easy way to estimate this data without overcomplicating the approach design, producing an acceptable result, and without imposing too many constraints on the input data. Therefore, this method can be used in everyday use cases.

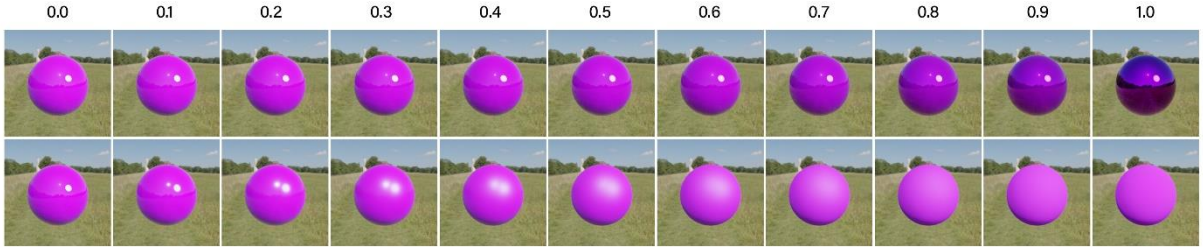


Figure 41: The first row shows changes in the metallic parameter, while the second shows the changes in roughness parameter in Blender. Overall, increasing the metallic parameter adds a mirror-like reflection to the object, while increasing the roughness removes specular reflections.

## IV.2. Dataset Generation

We created a new scene generator in Blender for reflectance estimation task. First, we randomly selected an object from one of the nine Blender primitives illustrated in Figure 33 and placed it on a plane. We then chose a random HDRI [URL4] to light the environment and a random texture [URL3] to apply to the table. We placed the camera in the scene at a random location while it focuses on the object. We then chose a random color for the object.

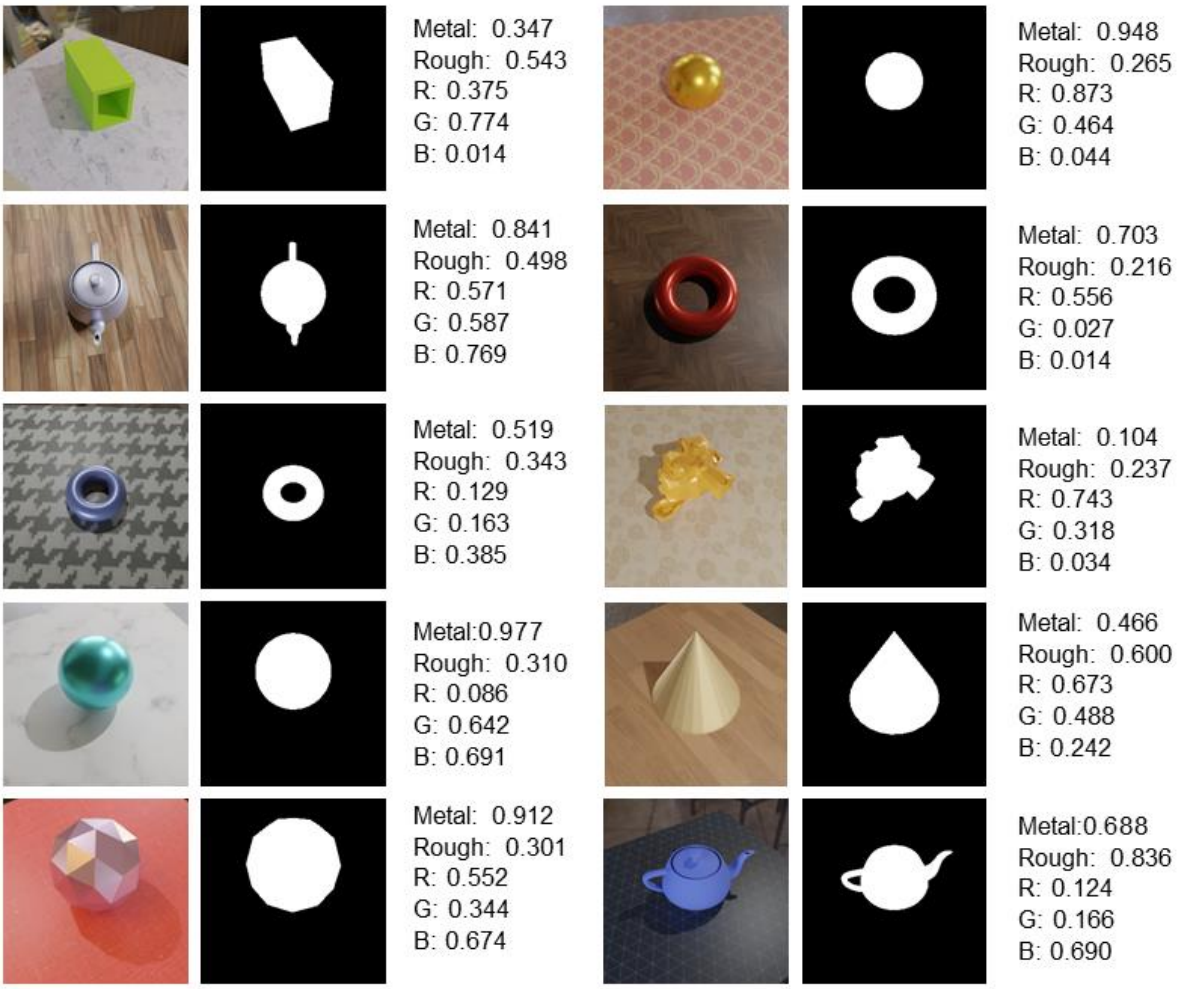


Figure 42: Some examples from our scene generator. For each scene, we generate a color image, a binary mask and a text file containing the object's Metallic, roughness and RGB values.

Blender used a BRDF model which consists of many parameters. We only changed the metallic and roughness parameters of each object and kept all other parameters at their initial value. Figure 41 illustrates how an object will look when these two parameters are altered. The first row shows changes in the metallic parameter, while the second shows the changes in roughness parameter. These parameters can get an arbitrary floating-point value in the range of zero and one. Overall, increasing the metallic parameter adds a mirror-like reflection to the object, while increasing the roughness removes specular reflections.

Eventually, we saved the color, metallic, and roughness values of the object in a text file and rendered each scene using Blender's Cycles. We then generated a binary

mask for the object. Figure 42 shows some examples generated by our scene generator.

## IV.3. Experiments

Since estimating material properties is a difficult problem, we considered using neural networks, which have demonstrated good performance on a variety of challenging tasks. We first started the experiment by training only one network, since it was not clear for us whether using only one network would be enough or whether we should take an indirect approach and use multiple models similar to the state-of-the-art. We defined the problem as a regression and used an inception-v4 [SIVA16] model. We changed the output layer to estimate five values of metallic, roughness and R, G, B. We added a sigmoid layer at the end to produce the values between 0 and 1.

We weren't sure which type of input to use when training the network, so we trained it twice, once with only the masked image (binary masked applied to the color image, thus 3 channel input) and once with both the masked image and the color image (6 channel input). We used Mean squared error as loss function (MSE). Following is the definition of the MSE:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \check{Y}_i)^2$$

Where $Y_i$ are the outputs from the network, $\check{Y}_i$ are the ground-truth values and N is the number of samples (batches). However, this formula is only valid for a network with a single output node. Since we had five output nodes in our case, we also had to calculate the average of loss over the output nodes:

$$MSE = \frac{1}{O * N}\sum_{i=1}^{N}\sum_{j=1}^{O}(Y_{ij} - \check{Y}_{ij})^2$$

Where in our case *o* is 5. We trained both networks on 10,000 synthetic training data and validated them on 100 synthetic images. We shuffled the training data at each epoch during training to prevent the network from learning the order of the data. Since our model was pretrained on ImageNet dataset, we normalized the input data using the method previously discussed in the section 3.3.1 for faster convergence. We also

used an Adam optimizer and trained the networks with the batch size of 8. Both networks converge around the 6$^{th}$ epoch with the MSE of 0.0237.

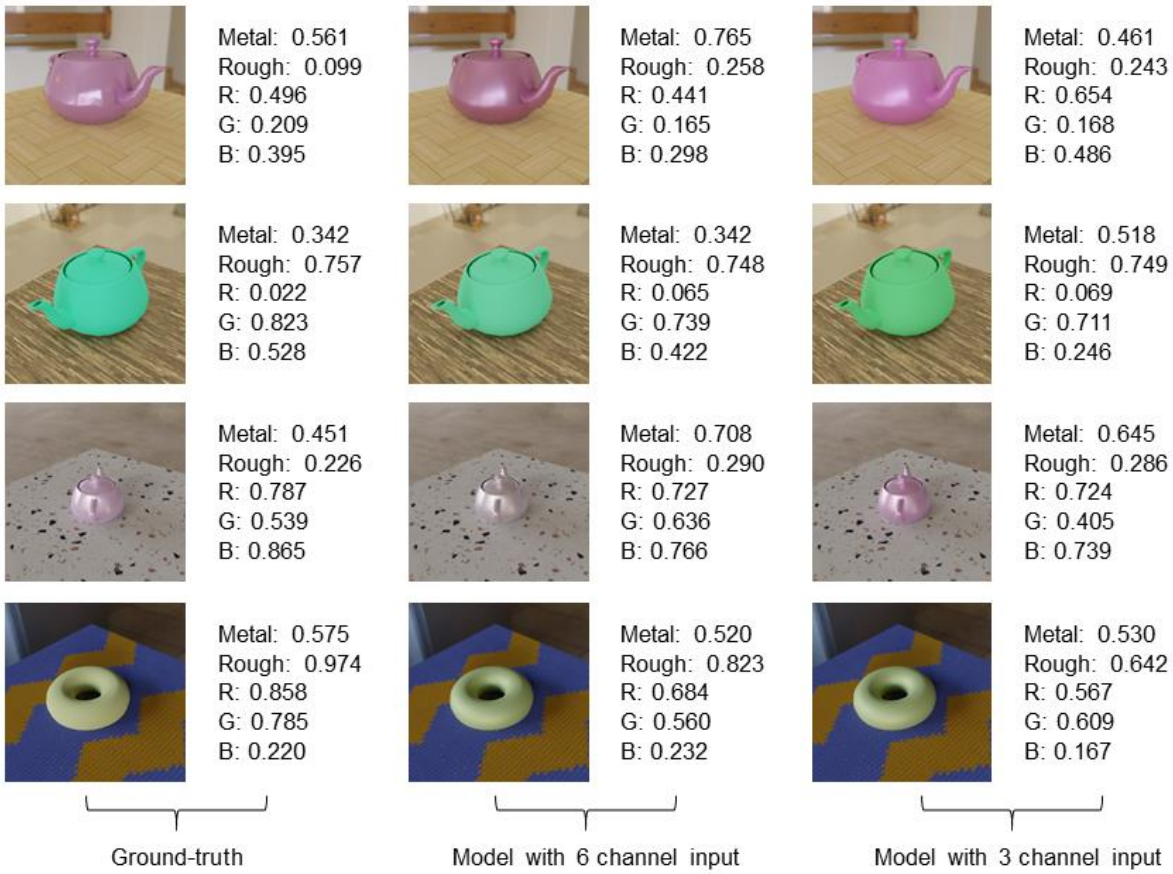


Figure 43: Results of the networks on synthetic test data. Although the results from both networks look similar, the result from the model trained with 6 channels input looks closer to the ground-truth, especially in the first and second rows.

We compared the results of both networks on synthetic data. Figure 43 shows this comparison. Although the results from both networks are similar, the result from the model trained with 6 channels input looks closer to the ground-truth, especially in the first and second rows.

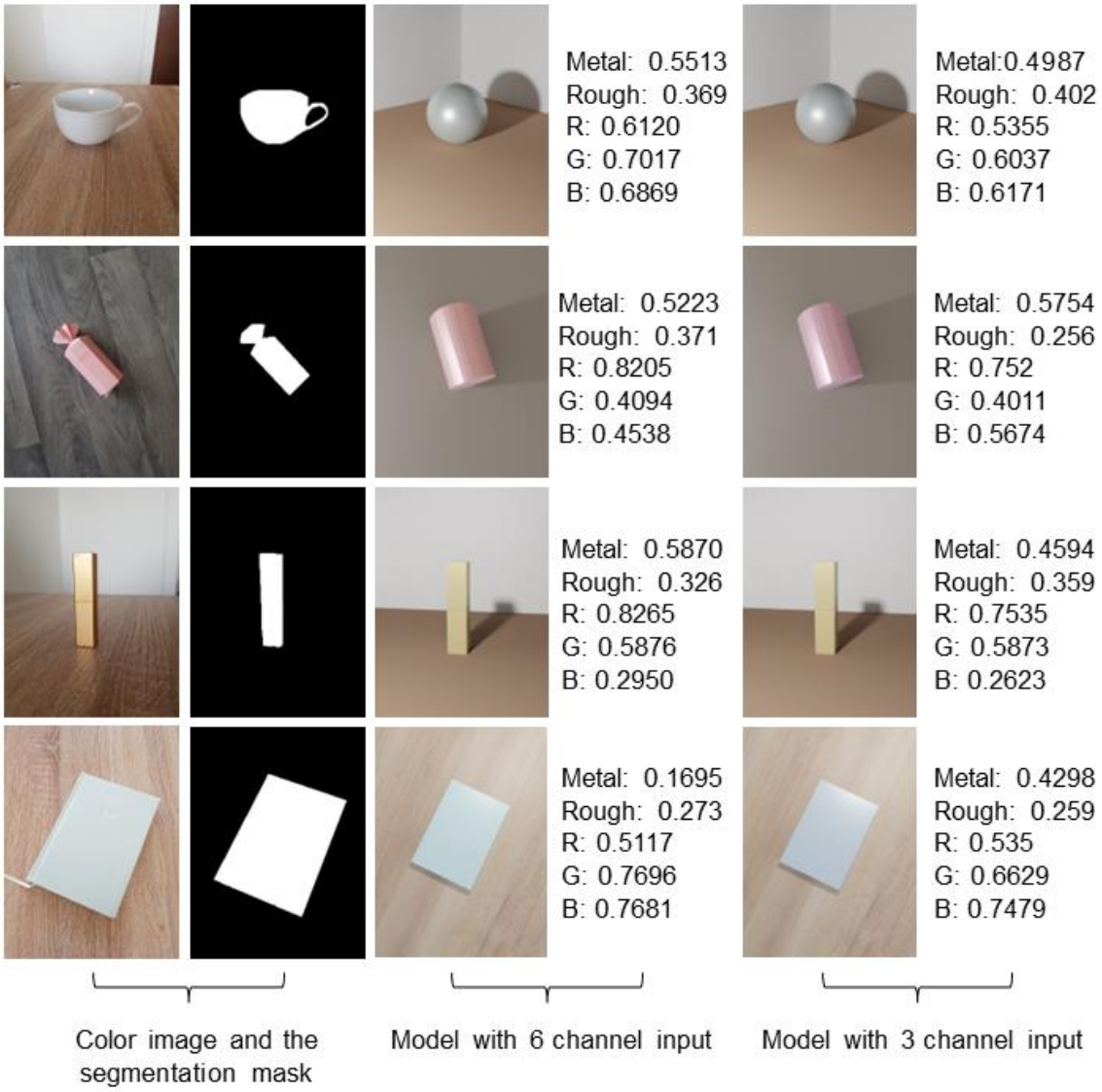


Figure 44: Results of our networks on real-world data. We used the output values of each network as the material values of an arbitrary primitive shape in Blender and rendered it using Cycles engine.

We also compared the results from the networks on real-world arbitrary objects. To create the real-world data, we placed a single object in a scene and took a photo of it with a mobile phone, and then manually created the corresponding segmentation mask. Figure 44 shows the results. We used the output values of each network as the material values of an arbitrary primitive shape in Blender and rendered it using Cycles engine. The color values of the objects appear to be close enough, but it was difficult to interpret the combination of metallic and roughness values.

As a result, we conducted another experiment; this time we removed the metallic parameter as it is mostly suitable for mirrors. We then added a point light at a random location near the camera to light up the object. This is because sometimes a specular object's shininess was not visible through the camera since the direction of the incoming light in HDRI lighting environments was random.

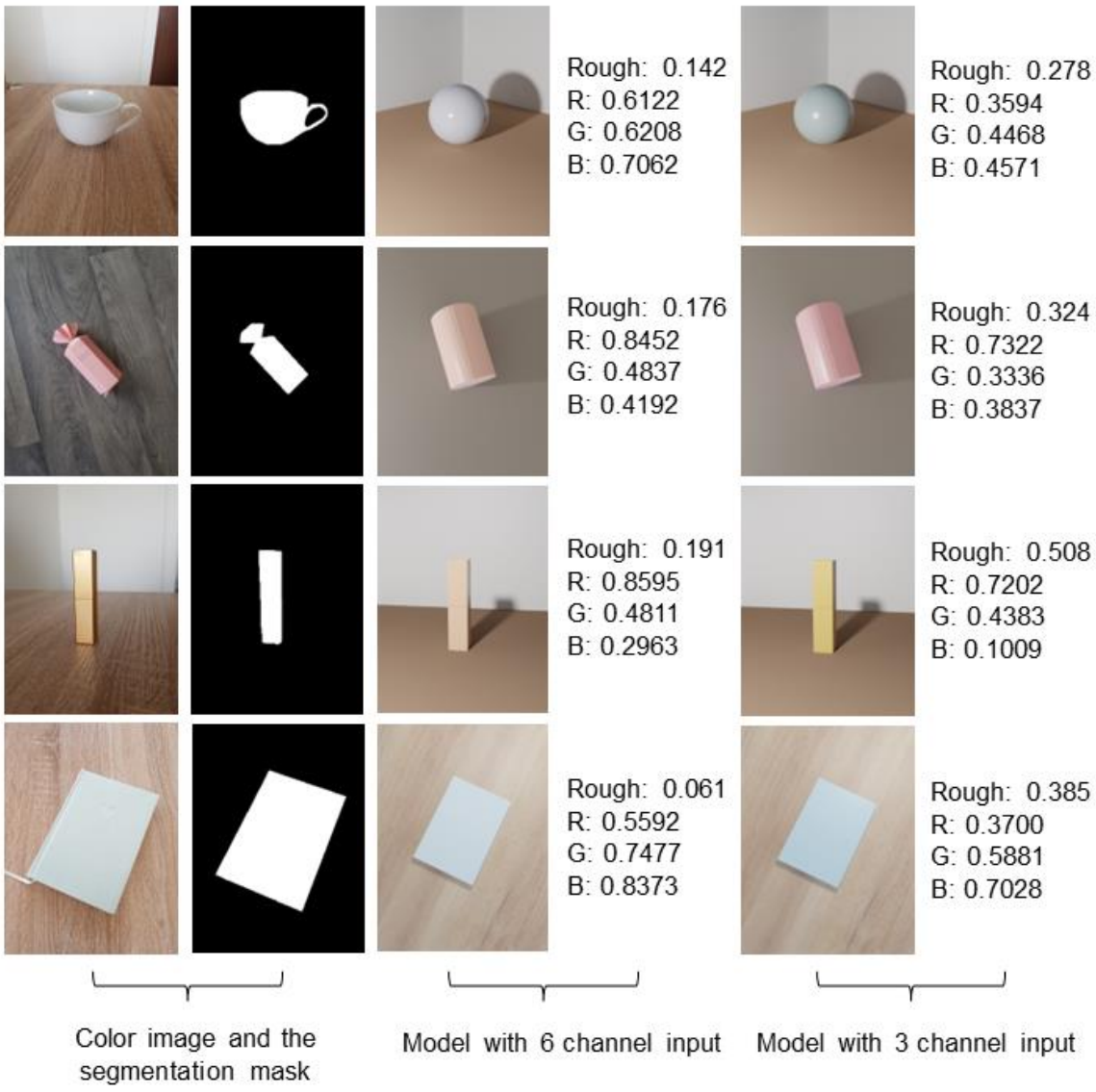


Figure 45: Results from our networks. For easier interpretation of the results, we removed the metallic parameter. However, there were not a drastic change in the output, meaning that roughness parameter alone could be enough for most items.

We generated 10,000 synthetic training data and 100 validation data again. We retrained both networks on the newly generated data. The network with 3 channel input converges around 5$^{th}$ epoch with MSE of 0.0207 and the network with 6 channel input converges around 6$^{th}$ epoch with MSE of 0.0176.

## IV.4. Preliminary Conclusion

In this chapter, we have proposed a preliminary study concerning the estimation of the material properties of an object of arbitrary shape, from a single image and under uncontrolled lighting conditions. The proposed method is simple and gives results that we consider acceptable given our initial objectives, but it is only a first attempt and needs to be further developed.

In the first experiment, the results seem to be correct, but the consideration of the ambient lighting remains a problem and makes the analysis of the results delicate. Also, the metallic and roughness values in the first experiment remain difficult to interpret and the removal of the metallic parameter in the second experiment did not bring any radical changes to the network results. Therefore, we will change the BRDF model to reduce the number of parameters to be determined. In general, one way to improve the performance of the network on real-world data could be to add more complex-looking objects in the synthetic training data.

.

# Chapter V. Conclusion

This thesis is about “Simultaneous acquisition of geometry and reflectance” using RGB-D cameras. In this work, we discussed this subject in two different sections: geometry acquisition and reflectance acquisition. Our main objective was to get both the geometry and material of an object using low-cost RGB-D cameras. These cameras are more accessible to the general public, but as previously discussed, the depth map obtained from them may contain holes or incorrect measurements. This is primarily an issue with active sensors and occurs on specular or transparent objects.

For our study of geometry acquisition, we first examined Intel Real Sense D435 camera and carefully observed its limitations in depth sensing. We then proposed a method based on deep learning to correct and complete the depth of close-range specular objects. Our method consists of four main parts:

- Dataset generation: We created a scene generator that generates synthetic dataset to train the networks with pixel-perfect ground-truth data. Each data consists of a color image with a corresponding normal map and a boundary map.
- Network training: We trained convolutional neural networks to estimate normal map and a boundary map from a single color image. We compared different network architectures to find the best one for our data. We used the output from these networks for the next two parts.
- Incorrect depth pixel removal: We located and removed the incorrect regions in the depth map in three steps: first, we removed the object boundaries using the boundary map generated by our network, second, we compared the normal

map estimated from the network and the normal map calculated from the depth map and removed the areas where these two differ significantly. At last, we removed all the remaining noise by using a morphological transformation.

- Depth completion: we completed the depth map by using an optimization approach and with the guidance of the outputs from normal estimation and boundary detection networks.

Following that, we demonstrate that our pipeline effectively improves the quality of the acquired depth map.

For the study of material acquisition, we investigated a straightforward method to estimate the material properties of an object from a single color image. Our proposed method consists of two steps:

- Dataset generation: We modified the scene generator that we previously created in geometry acquisition section and created a new synthetic dataset. Each data consists of a color image, a binary mask and a text file containing the object's Metallic, roughness and RGB values.
- Network training: We trained a neural network model on our synthetic data to directly estimate the material properties.

The network generates decent result, but more experiments are required to improve its performance.

## V.1. Future work

Regarding geometry acquisition, in chapter 3 discussion section, we explained an uncommon drawback that occurs when locating and removing the incorrect depth. This drawback can get investigated in future research. In addition, another interesting subject would be to investigate the impact of noise differentials of synthetic data compared to real-world data in our pipeline.

Regarding material estimation, the experiments that we presented are still in their early stages. There are many ways in which the results can be improved. One way is to thoroughly examine the synthetic dataset to ensure it has enough diversity regarding its object shapes and shadow positions to cover all real-life situations.

Additionally, since we are using RGB-D cameras, we have both the color image and the corresponding depth map in a single acquisition. Thus, we can use them both for improving the material estimation. As stated previously, the depth map obtained by active RGB-D sensors usually suffers from noise, missing values, and incorrect

measurements on specular and transparent objects. Analyzing the depth map thoroughly for types of noise and missing values can provide valuable information about the material properties and the light direction. Furthermore, the shape of the object has an important role in the color image formation process, adding this information can help improve the results as well.

# Bibliography

[BCCN18] Bianco, S., Cadene, R., Celona, L., & Napoletano, P. (2018). Benchmark Analysis of Representative Deep Neural Network Architectures. IEEE Access, 6, 64270–64277. https://doi-org.ezproxy.unilim.fr/10.1109/ACCESS.2018.2877890

[BKPB17] Bonneel, N., Kovacs, B., Paris, S., & Bala, K. (2017). Intrinsic Decompositions for Image Editing. In Computer Graphics Forum (Proc. Eurographics STAR), 36(2), 593–609. https://doi.org/10.1111/cgf.13149

[DADDB19] Flexible SVBRDF Capture with a Multi-Image Deep Network. Deschaintre V., Aittala M., Durand F., Drettakis G., Bousseau A. Computer Graphics Forum (EGSR Conference Proceedings), 38, 4(July 2019)

[CDF17] Chang, A., Dai, A., Funkhouser, T., Halber, M., Niebner, M., Savva, M., Song, S., Zeng, A., & Zhang, Y. (2017). Matterport3D: Learning from RGB-D Data in Indoor Environments. *2017 International Conference on 3D Vision (3DV), 3D Vision (3DV), 2017 International Conference on, 3DV*, 667–676. https://doi-org.ezproxy.unilim.fr/10.1109/3DV.2017.00081

[CPKMY17] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 40(4), 834–848. https://doi.org/10.1109/TPAMI.2017.2699184

[CSS16] Chakrabarti, A., Shao, J., & Shakhnarovich, G. (2016). Depth from a Single Image by Harmonizing Overcomplete Local Network Predictions. *Conference on Neural Information Processing Systems (NIPS) 2016, Barcelona.*

[CPSA17] Chen, L.-C., Papandreou, G., Schroff, F. & Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation. ArXiv, abs/1706.05587.

[DDSLLF09] Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, Li. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition, Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference On*, 248–255. https://doi-org.ezproxy.unilim.fr/10.1109/CVPR.2009.5206848

[EF15] Eigen, D., & Fergus, R. (2015). Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2650–2658. https://doi.org/10.1109/ICCV.2015.304

[FGW20] Fu, L., Gao, F., Wu, J., Li, R., Karkee, M., & Zhang, Q. (2020). Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review. Computers & Electronics in Agriculture, 177, N.PAG. https://doi.org/10.1016/j.compag.2020.105687

[GGGDG16] Guarnera D., Guarnera G. C., Ghosh A., Denk C., Glencross M.: BRDF Representation and Acquisition. *Computer Graphics Forum (2016).*

[GRCL22] Garces, E., Rodriguez-Pardo, C., Casas, D., & Lopez-Moreno, J. (2022). A Survey on Intrinsic Images: Delving Deep into Lambert and Beyond. *International Journal of Computer Vision*, 1–33. https://doi-org.ezproxy.unilim.fr/10.1007/s11263-021-01563-8

[GRR18] Georgoulis, S., Rematas, K., Ritschel, T., Gavves, E., Fritz, M., Van Gool, L., & Tuytelaars, T. (2018). Reflectance and Natural Illumination from Single-Material Specular Objects Using Deep Learning. in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1932-1947, 1 Aug. 2018, doi: 10.1109/TPAMI.2017.2742999.

[GSB18] Gomes, L., Silva, L. & Bellon, O. (2018). Exploring RGB-D Cameras for 3D Reconstruction of Cultural Heritage: A New Approach Applied to Brazilian Baroque Sculptures. Journal on Computing and Cultural Heritage. 11. 1-24. 10.1145/3230674.

[HHC14] Huang, Y.-L., Hsu, T.-W., & Chien, S.-Y. (2014). Edge-aware depth completion for point-cloud 3D scene visualization on an RGB-D camera. *2014 IEEE Visual Communications and Image Processing Conference, Visual Communications and Image Processing Conference, 2014 IEEE*, 422–425. https://doi.org/10.1109/VCIP.2014.7051596

[HKMT90] Holschneider, M., Kronland-Martinet, R., Morlet, J., & Tchamitchian, P. (1990). A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform. Springer Berlin Heidelberg. https://doi-org.ezproxy.unilim.fr/10.1007/978-3-642-75988-8_28

[HS79] Horn, B. K. P., & Sjoberg, R. W. (1979). Calculating the reflectance map. Applied Optics, 18(11), 1770–1779.

[HZRS16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference On*, 770–778. https://doi-org.ezproxy.unilim.fr/10.1109/CVPR.2016.90

[IR22] Intel RealSense D400 Series Product Family Datasheet – 2022, url: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet

[IS15] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *In Proceedings of the 32nd International Conference on International Conference on Machine Learning* - Volume 37 (ICML'15). JMLR.org, 448–456.

[KK11] Krähenbühl, P., & Koltun, V. (2011). Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials.

[KSH12] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems.* 25. 10.1145/3065386.

[LGL12] Junyi Liu, Xiaojin Gong, & Jilin Liu. (2012). Guided inpainting and filtering for Kinect depth maps. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Pattern Recognition (ICPR), 2012 21st International Conference On*, 2055–2058.

[LSD15] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965.

[LZP14] Ladicky, L., Zeisl, B., & Pollefeys, M. (2014). Discriminatively Trained Dense Surface Normal Estimation. *In 2014 ECCV.*

[MMZ18] Meka, A., Maximov, M., Zollhofer, M., Chatterjee, A., Seidel, H.-P., Richardt, C., & Theobalt, C. (2018). LIME: Live Intrinsic Material Estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2018, pp. 6315-6324, doi: 10.1109/CVPR.2018.00661.

[MS90] Murray-Coleman J.F. & Smith A.M. (1990) The Automated Measurement of BRDFs and their Application to Luminaire Modeling. In *Journal of the Illuminating Engineering Society*, 19:1, 87-99, DOI: 10.1080/00994480.1990.10747944

[NRH77] Nicodemus, F.E., Richmond, J.C., Hsia, J.J., Ginsberg, I.W., & Limperis, T. (1977). Geometrical considerations and nomenclature for reflectance. In: Monograph, vol. 161. National Bureau of Standards (US).

[P19] Iakubovskii, P. (2019) Segmentation Models Pytorch, Github, URL: https://github.com/qubvel/segmentation_models.pytorch

[P75] Phong, B. (1975) Illumination for computer generated pictures. Communications of the ACM,18(6):311–317, June 1975, DOI: https://doi.org/10.1145/360825.360839

[PBN16] Paul, S., Basu, S., & Nasipuri, M. (2016). Microsoft Kinect in Gesture Recognition: A Short Review.

[RDSK15] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. & Li, F. (2015) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **115,** 211–252. https://doi.org/10.1007/s11263-015-0816-y

[RFB15] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.

[SGHG21] Souley Dosso, Y., Greenwood, K., Harrold, J., & Green, J. R. (2021). RGB-D scene analysis in the NICU. Computers in Biology and Medicine, 138. https://doi.org/10.1016/j.compbiomed.2021.104873

[SIVA16] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. AAAI Conference on Artificial Intelligence. 31. 10.1609/aaai.v31i1.11231.

[SLJS14] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.*

[SMPN20] Sajjan, S., Moore, M., Pan, M., Nagaraja, G., Lee, J., Zeng, A., & Song, S. (2020). Clear Grasp: 3D Shape Estimation of Transparent Objects for Manipulation. *2020 IEEE International Conference on Robotics and Automation (ICRA), Robotics and Automation (ICRA), 2020 IEEE International Conference On*, 3634–3642. https://doi.org/10.1109/ICRA40945.2020.9197518

[SVISW16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308.*

[SZ14] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR 2015*

[T04] Telea, A. (2004). An Image Inpainting Technique Based on the Fast Marching Method. *Journal of Graphics Tools*, *9*(1), 23–34. https://doi.org/10.1080/10867651.2004.10487596

[TFT20] Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin, B. R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J. -Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., & Zollhöfer, M. (2020). State of the Art on Neural Rendering. *Computer Graphics Forum*, *39*(2), 701–727. https://doi.org/10.1111/cgf.14022

[TS67] Torrance, K.E., & Sparrow, E.M. (1967). Theory for off-specular reflection from roughened surfaces. *J. Optical Soc. America,* 57:1105–1114.

[URL1] Stanford 2017 Lectures: http://cs231n.stanford.edu/slides/2017/

[URL2] Intel Realsense SDK2 sample program, url: http://unanancyowen.com/en/realsense-sdk-2-samples/

[URL3] url: https://resources.blogscopia.com/category/textures/, url: https://3dtextures.me/, and url: https://ambientcg.com/

[URL4] Poly Haven: The Public 3D Asset Library, url: https://polyhaven.com/hdris

[URL5] Intel RealSense SDK 2.0, url: https://www.intelrealsense.com/sdk-2/

[URL6] Poly Haven Models, url: https://polyhaven.com/models

[URL7] TaraXL sensor by econ systems, url: https://www.e-consystems.com/3d-usb-stereo-camera-with-nvidia-accelerated-sdk.asp

[URL8] Matterport pro3, url: https://matterport.com/fr/pro3#pro3-packages

[URL9] Kinect adventures game, url: https://www.xboxpassion.fr/jeux-xbox-360/336-kinect-adventures.html

[URL10] Xbox Kinnect in Sunnybrook hospital, url: https://www.youtube.com/watch?v=f5Ep3oqicVU

[WFG15] Wang, X., Fouhey, D. F., & Gupta, A. (2015). Designing deep networks for surface normal estimation. *2015 7th International Conference on Games & Virtual Worlds for Serious Applications (VS-Games)*, 539–547. https://doi.org/10.1109/CVPR.2015.7298652

[YPCLY16] Yang, J., Price, B., Cohen, S., Lee, H., & Yang, M.-H. (2016). Object Contour Detection with a Fully Convolutional Encoder-Decoder Network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 193–202. https://doi-org.ezproxy.unilim.fr/10.1109/CVPR.2016.28

[ZF18] Zhang, Y., & Funkhouser, T. (2018). Deep Depth Completion of a Single RGB-D Image. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Computer Vision and Pattern Recognition (CVPR), 2018 IEEE/CVF Conference on, CVPR*, 175–185. https://doi-org.ezproxy.unilim.fr/10.1109/CVPR.2018.00026

[ZJL20] Zhao, S., Jakob, W. & Li, T.M: Physics-Based Differentiable Rendering: From Theory to Implementation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Courses (SIGGRAPH '20 Courses),* August 17, 2020. ACM, New York, NY, USA, 30 pages. https://doi.org/10.1145/3388769.3407454

[ZS17] Zhang, Y., Song, S., Yumer, E., Savva, M., Lee, J.-Y., Jin, H., & Funkhouser, T. (2017). Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

**Simultaneous Acquisition of Geometry and Reflectance**

Cette thèse porte sur « l'acquisition simultanée de la géométrie et de la réflectance ».
Les caméras RGB-D sont utilisées dans une variété d'applications, y compris la numérisation 3D, la navigation et la manipulation de robots, etc. Ces caméras fournissent simultanément une image couleur (RVB) et une carte de profondeur. La carte de profondeur indique la distance entre les objets et la caméra par pixel. Ces caméras RGB-D sont disponibles dans une variété de prix ; avec en particulier des modèles low-costs destinés à un usage grand public. Cependant, ces caméras présentent des défauts de mesure dans certaines zones / sous certaines conditions. Par conséquent, la carte de profondeur obtenue présente des valeurs manquantes (trous), des mesures de profondeur incorrectes et du bruit. Ces problèmes sont plus fréquents dans les zones où les objets sont transparents, spéculaires, trop proches ou trop éloignés, ou trop fins.
Dans ce travail, nous proposons une approche pour corriger et compléter la carte de profondeur d'objets spéculaires proches. Notre approche se compose de différentes parties : d'abord, nous créons un générateur de scènes 3D aléatoires qui génère des images synthétiques pour l'entraînement du réseau de neurones. Deuxièmement, nous entraînons des réseaux de neurones à l'aide de nos images synthétiques pour aider à identifier les régions incorrectes dans la carte de profondeur. Finalement, nous complétons la profondeur en utilisant une méthode d'optimisation. Nous testons notre pipeline proposé sur des données du monde réel et démontrons qu'il obtient d'excellents résultats.
Obtenir la réflectance d'un objet réel à partir d'une seule image est une tâche difficile. Nous proposons une première approche simplifiée par une adaptation de la méthode que nous avons proposée précédemment : création d'un nouveau générateur de scènes pour générer des ensembles de données synthétiques et entraîner des réseaux de neurones. L'objectif est d'obtenir une première approximation utilisable par un graphiste.

Mots-clés : Complétion de la profondeur, Images RGB-D, Images synthétiques

**Simultaneous Acquisition of Geometry and Reflectance**

This thesis focuses on the “simultaneous acquisition of geometry and reflectance”.
RGB-D cameras are used in a variety of applications, including 3D scanning, robot navigation and manipulation, and so on. These cameras provide a color (RGB) image and a depth map simultaneously. The depth map indicates the distance between objects and the camera per pixel. These RGB-D cameras are available in a variety of prices; with the low-cost models intended for general public use. However, these low-cost cameras suffer from measurement errors in certain areas / under certain conditions. Therefore, the result depth map contains missing values (holes), incorrect depth measurements, and noise. These issues are most common in areas where objects are transparent, specular, too close or too far away, or too thin. In this work, we propose an approach for correcting and completing the depth of close-range specular objects. Our approach consists of several steps: First, we create a random 3D scene generator that generates synthetic images for training neural network. Second, we train neural networks using our dataset to help identifying incorrect regions in the depth map. Eventually, we complete the depth using an optimization method. We test our proposed pipeline on real-world data and demonstrate that it achieves excellent results.
Obtaining an object's material properties (reflectance) from a single image is a challenging and ill-posed task. We propose an early approach simplified by an adaptation of the method we proposed previously: creation of a new scene generator to generate synthetic datasets and train neural networks. The objective is to obtain a first approximation usable by a graphic designer.

Keywords: Depth completion, RGB-D images, Synthetic dataset