UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Bilel Bennadji

USING ENCODER-DECODER ARCHITECTURE FOR MATERIAL SEGMENTATION BASED ON BEAM PROFILE ANALYSIS

Master's Thesis
Degree Programme in Computer Science and Engineering
July 2020

Bennadji B. (2020) Using Encoder-Decoder Architecture for Material Segmentation Based on Beam Profile Analysis. University of Oulu, Degree Programme in Computer Science and Engineering, 63 p.

ABSTRACT

Recognition and segmentation of materials has proven to be a challenging problem because of the wide divergence in appearance within and between categories. Many recent material segmentation approaches treat materials as yet another set of labels like objects. However, materials are basically different from objects as they have no basic shape or defined spatial extent. Our approach roughly ignores this and can primarily take advantage of limited implicit context (local appearance) as it seems during training, because our training images that almost do not have a global image context; such as (I) where the used materials have no inherent shape or defined spatial extent like apple, orange and potato approximately have the same spherical shape; (II) besides, images where taken under a black background, which roughly removes the spatial features of the materials.

We introduce a new materials segmentation dataset, which was taken with a Beam Profile Analysis sensing device. The dataset contains 10 material categories, and it has image pair samples consisting of grayscale images with and without the laser spots (grayscale and laser images) in addition to annotated segmented images. To the best of our knowledge, this is the first material segmentation dataset for Beam Profile Analysis images.

As a second step, we proposed a deep learning approach to perform material segmentation on our dataset; our proposed CNNs is an encoder-decoder model, which is based on the DeeplabV3+ model. Our main goal is to obtain segmented material maps and discover how the laser spots contribute to the segmentation results; therefore, we perform a comparative analysis across different types of architectures to observe how the laser spots contribute to the whole segmentation. We built our experiments on three main types of models that use a different type of input; for each model, we implemented various types of backbone architectures. Our experiments results show that the laser spots have an efficient contribution on the segmentation results. GrayLaser model achieves a significant accuracy improvement compared to other models, where the fine-tuned architecture of this model has reached an accuracy of 94% over MIoU metric, and one trained from the scratch has reached an accuracy of 62% over MIoU.

Keywords: semantic segmentation, deep learning, convolutional neural networks, laser spots, grayscale images, data annotation.

TABLE OF CONTENTS

FOREWORD

I would like to acknowledge the support and encouragement I received during my thesis research. I am incredibly grateful for those of you who shared with me personal and professional supports. I thank you for helping me during the journey. First, I want to thank my supervisor the Professor Janne Heikkilä. He has taught me how to become a good researcher at work. I appreciate his contributions of time, ideas and letting me work on my ideas as well during the whole of this work.

Second, I want to thank my colleagues and my friends in the Center for Machine Vision and Signal Analysis (CMVS). Thank all of you for providing support, mentoring, and motivation along the way.

Finally, my most sincere gratitude goes to my family. I give my heartfelt thanks to my parents. I cannot be here without their unlimited supports and encouragements.

Oulu, June 16, 2020.

Bilel Bennadji

LIST OF ABBREVIATIONS AND SYMBOLS

3D	three dimensional
VR	Virtual Reality
AR	augmented reality
ToF	Time of Flight
SL	Structured Light
LIDAR	Light Detection And Ranging
RGB-D	Red-Green-Blue-Depth
IR	Infrared
BPA	Beam Profile Analysis
CV	Computer Vision
FoV	field of view
fps	Frame per second
DL	Deep Learning
ML	Machine learning
2D	two dimensional
CNN	convolutional neural networks
RNN	recurrent neural networks
GAN	generative adversarial networks
VGA	Video Graphics Array
ANN	artificial neural networks
SVM	Support Vector Machine
DBN	deep belief network
SAE	tacked auto encoders
CONV	convolutional layer
POOL	pooling layer
FC	fully connected layer
ReLU	rectified linear unit
FCN	fully convolutional network
ASSP	atrous spatial pyramid pooling
BT	batch normalization
OS	output stride
DoF	degree of freedom

1. INTRODUCTION

3D imaging solutions can be divided into passive and active technologies. Passive sensing relies on scene details such as corners, edges and points that are uniquely distinguishable from conventional RGB images. Common passive sensing methods include stereo, structure from motion, and depth from focus. The main weakness of the passive methods is that they can only provide 3D information from locations where details are visible.

In contrast, active 3D sensing solutions use light patterns projected to the scene that enable denser and more regular sampling even from surfaces that do not have any details. The most common approaches used in active depth cameras include structured light (SL) and time of flight (ToF).

In this regard, Beam Profile Analysis (BPA) sensing solution is an active depth measuring technology based on SL technique. It uses a grid of laser beams projected to the scene and a CMOS sensor for observing the reflected beams. The sensor produces image pairs consisting of grayscale images with and without the laser spots, where those laser spots have important information about the depth and the material information of the scene. Therefore, they allow classifying materials based on specific scattering/reflection properties.

Convolutional neural networks (CNN) have recently shown to provide great potential for various image prediction problems including depth estimation and semantic segmentation. The previous works on CNNs form a solid basis to develop a learning-based approach for smart image interpolation and material segmentation using the features of laser spots produced by the BPA sensing solution.

The objective of semantic segmentation is to assign a class to each pixel describing what type of object it belongs to. Similarly, for material segmentation, each pixel is assigned a material class. Recognizing materials is essential for understanding, interacting, and summarizing complex and different scenes. Material recognition plays a significant role in many applications.

Our project aims to devise an alternative interpolation scheme based on deep learning and scene understanding, which delivers a material-based segmentation of the scene by using laser images as well as the grayscale images based on beam profile analysis. The work starts with creating a material segmentation dataset observed through the BPA sensing solution including annotation of all relevant materials. As a second step, the proposed network architectures were implemented and trained with the goal of getting a segmented material map of the scene.

2. DEPTH ESTIMATION

The human visual system is designed for depth perception, which is the ability to see things in three dimensions (3D), including width, length, and depth. This perception is possible through a combination of different physiological and psychological structures and functions. Generally depth perception needs to have a binocular (two eyes) vision. It is called stereopsis when the pair of eyes look at an object from slightly different angles and the brain processes and compare two sets of information to form only one image effectively [1]. Sometimes the brain recognizes depth primarily using one eye that is called monocular vision. Some of the monocular cue structures are relative size, familiar size, motion parallax, occultation, and differences in brightness [1].

Depth estimation or extraction of the depth refers to the set of methods and algorithms aiming to obtain a three-dimensional representation of the spatial structure of a scene, in other words, to get the distance measurements of each point of the viewed scene. In the computer vision field, depth estimation from scenes or objects has been studied for a long time and it is a crucial and fundamental problem. Depth estimation has been applied in various applications to make them more feasible, robust and to make machines identify objects, and not only for capturing the object but also to understand the representation of the objects because representation is the main step in recognition. Some of these applications include 3D modelling, semantic segmentation, computer graphics, virtual reality (VR), and augmented reality (AR).

In general terms, we can divide all the methods to obtain the depth estimate from the real-world scene as active and passive depth sensing techniques, both of which are extremely popular [2][3]. Some of the methods to estimate depth use the human visual system as an essential source of inspiration when a set of algorithms are implemented to process videos or images in a complex manner by using several binocular and monocular cues. Other methods are sensor-based methods using time of flight like RGB-D cameras and LIDAR, and structured light (SL) like Microsoft Kinect.

In this chapter, a general review of the main techniques and technologies used for depth estimation is provided, presenting some hardware used to measure the depth and finally a general comparison of the techniques and devices.

2.1. Active Depth Sensing Techniques

The active techniques put some energy onto the target scene, by projecting it to illuminate the space and processing the reflected energy. Some of their advantages are that their depth accuracy is higher than the passive methods. However, they require energy compared to the passive methods. Some of the popular active methods are the time of flight (ToF) and structured light (SL).

2.1.1. Time of Flight

The term time of flight (ToF) refers to methods that implement the measurement of the distance using the phase delay of the adjusted light source received at different distance

and calculating the depth according to the speed of light as shown in Figure 1 [4], so the distance D can be calculated as follows:

$$D = \frac{c}{2f} \cdot \frac{\psi}{2\pi} \tag{1}$$

where f is the signal's frequency, c is the light's speed, and ψ is the phase difference between the emitted and reflected infrared (IR) signals. The attractive features of the ToF technique include its low cost, small size and video-rate depth data collection. However, there are some disadvantages of the ToF system including the need for active illumination synchronization, distance aliasing and the possibility for multi-path interference. This 3D sensing technology is used in some applications such as 3D reconstruction, mapping and object detection [5].

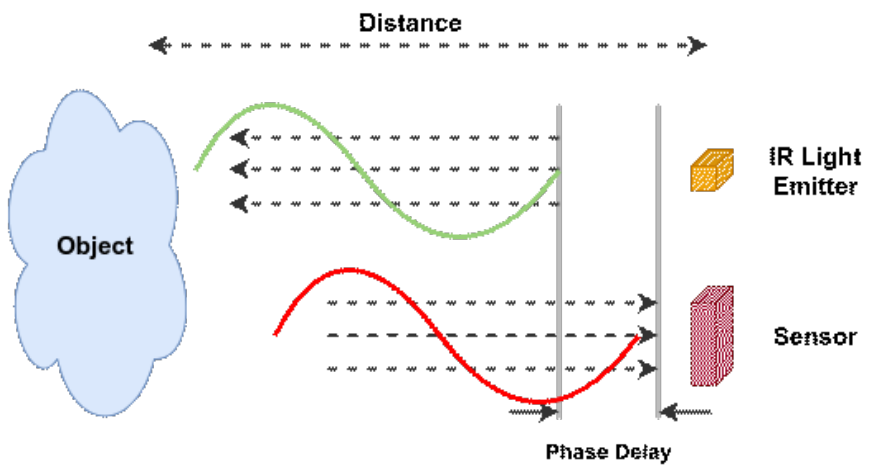

Figure 1. The principle of Time of Flight.

2.1.2. Structured Light

The principle of Structured Light (SL) technique is transmitting a predetermined light pattern to the object surface and to obtain the depth information based on the analysis of the distorted pattern [6]. The process of SL has two steps. Firstly, a laser projector is used to project the encoded beam onto a target object in order to generate the feature points. Then, depending on the projection model and the geometric pattern of the projected light, the triangulation principle is used to calculate the distance between the feature points and the optical centre of the camera, which allows generating the depth of the feature point and implement the reconstruction of the model. The encoded beam can represent the Structure Light, including various patterns such as points and lines.

Figure 2 shows the basic principle of the SL method. Usually the camera is described as a perspective projection model, and the corresponding relationship between object space and image plane can be expressed as:

$$s[u_c v_c 1]^T = M_c[X_w Y_w Z_w]^T \tag{2}$$

where s an arbitrary scale factor; $[u_c,v_c,1]$ are the homogeneous coordinates of a point P in the image coordinate system of the camera; M_c is the linear transformation matrix of 3x4; and $[X_w,Y_w,Z_w,1]$ are the homogeneous coordinates of a point P in the object world coordinate system. The projector can be seen as an inverse camera. Hence, the projector has a similar model as equation (2), where s^* is an arbitrary scale factor; $[u_p,v_p,1]$ are the coordinates of a point P in the image coordinate system of the projector; M_p is the linear transformation matrix of 3x4. Therefore, the 3D coordinates $[X_w,Y_w,Z_w,1]$ of the calculated point can be obtained by equations (2) and (3) with the image coordinates u_c,v_c and u_p,v_p with known values of M_c and M_p.

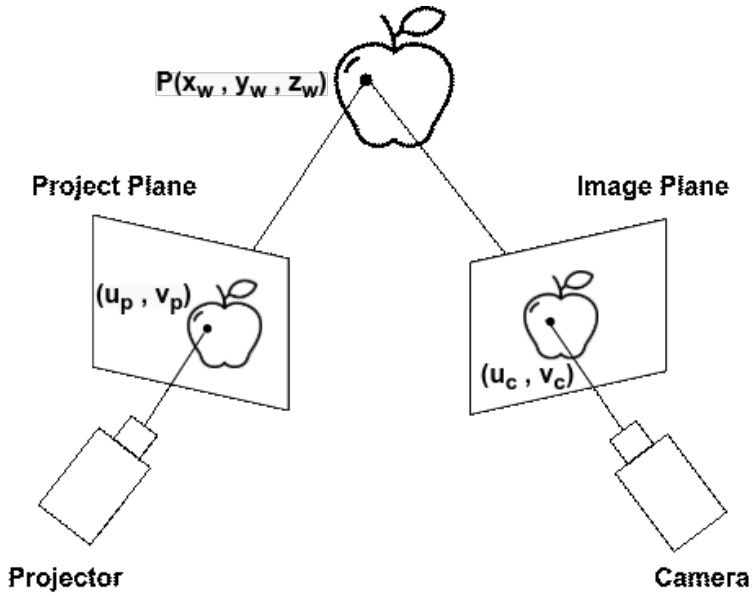

Figure 2. The basic principle of 3D SL method.

Due to the ease of implementation and high precision, the SL method is used in a vast range of applications such as object detection [7], 3D reconstruction [8] and scene understanding [9]. There are several hardware devices based on SL technology such as Kinect from Microsoft and RealSense from Intel. The next sections present brief descriptions for Kinect from Microsoft and Beam Profile Analysis devices.

Microsoft Kinect

Microsoft Kinect is an RGB-D sensor (depth sensor, a colour camera) providing synchronised colour and depth images. It was initially utilised as an input device for Xbox game consoles [10]. With a full-body 3D motion capture algorithm and facial recognition capabilities, Kinect allows interaction between players and a game without touching the controller. In recent years, the computer vision (CV) community discovered that Kinect's depth sensing technology could be extended far beyond the gaming industry and at a lower cost than conventional 3D cameras like TOF cameras and stereo cameras. Besides, Kinect opens a new solution for classical problems of CV by providing complementary nature of the colour and depth information.

Furthermore, a large number of research papers have been already presented in several CV conferences and journals, which clearly show the potential impact of Kinect

in the computer vision area after it was released [11] [12]. Some of those research papers include topics in object tracking, recognition and indoor 3D mapping.

Figure 3 shows the arrangement of a Kinect Sensing Hardware, consisting of an IR projector, an IR camera, and a colour RGB camera. The depth sensor includes the IR camera and the IR projector together to create the depth map where the IR projector emits an IR speckle dot pattern into the 3D scene while the reflection of the IR speckles is captured by the IR camera. The depth sensor has a practical range limit of 0.8 m to 3.5 m length and 30 fps for the outputs video with a resolution of 640 x 480 pixels. It has a field of view (FoV) of 57 degrees horizontally and 43 degrees vertically. While the RGB Camera operates at 30 Hz and has a resolution of 640 x 480 pixels with an 8-bit per channel, it has a resolution of 1280 x 1024 pixels as an additional option running at 10 fps.

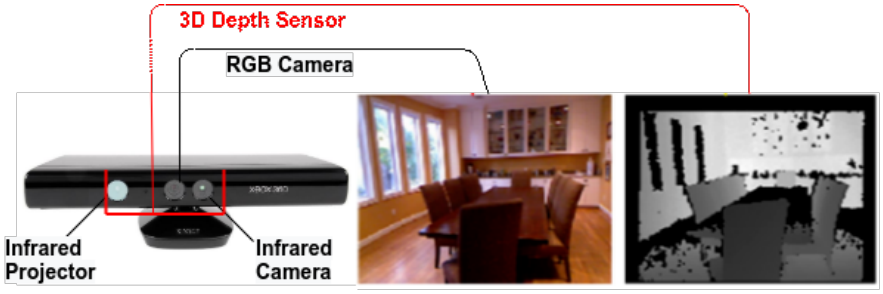

Figure 3. Kinect Sensing Hardware arrangement with two images captured by the depth camera and the RGB camera.

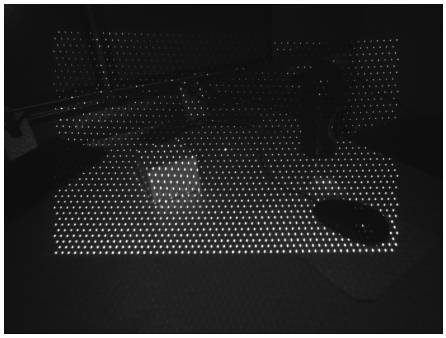

Figure 4. BPA technology spot image of a planar surface.

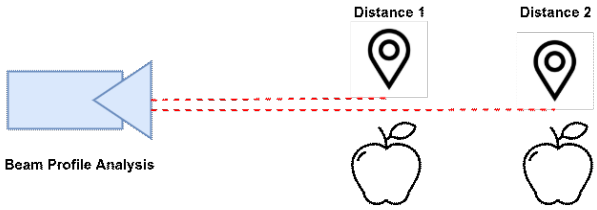

Figure 5. The basic principle of BPA.

Beam Profile Analysis technology

Beam Profile Analysis (BPA) technology is an active depth measuring technology based on SL technique. It projects a laser beam (light source) resulting in a matrix of regular dots pattern on the object to be measured, as Figure 4 shows. The reflection of each laser spot is captured by a CMOS camera, and its beam profile is then being analyzed where the 3D coordinates of these spots are computed [13][14], as Figure 5 shows. BPA technology can be used for a variety of applications such as in face authentication solutions in smartphones, segmentation for AR/VR purposes, image enhancements or object detection like what is used in bin-picking robots.

2.2. Passive Depth Sensing Techniques

Passive techniques are the techniques that deal with the natural ambient light and the extract optical information from the captured image without consuming energy compared to the active methods. They have only image sensors that capture the images and their algorithm of measuring the depth is divided into monocular and stereo cues.

2.2.1. Monocular Cue

Monocular Depth Estimation (MDE) is estimating the depth of a scene from a single image, which is considered a complex task for computational models to obtain high accuracy and with low resource requirements. Some of the advantages to be able to estimate depth from a single image is recovering depth information when other information such as stereo images, or optical flow is unavailable.

This is an easy task for humans because we can exploit features such as perspective, occlusion, relative, familiar size to known objects, lighting and shading, and more. Monocular depth estimation is an ill-posed problem because a single 2D image may be generated from an infinite number of distinct 3D scenes. Therefore, previous approaches have been published to exploiting the same statistically meaningful monocular cues of human vision such as perspective, occlusions, object sizes, and object localization. Other approaches have used multiple images (a series of 2D image sequences) that provide geometric constraints to overcome the ambiguities of photometric data. Some of those approaches include structure from motion, depth from focus, depth from defocus methods [15].

In recent years, with the rapid development in DL, deep neural networks have shown their excellent performance on computer vision, like image classification, semantic segmentation and objective detection. Furthermore, recent studies have shown that the pixel level monocular depth estimation task is promising in end-to-end learning based on DL. Various end-to-end deep neural networks have proved their effectiveness to address the MDE, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs) [16][17][18].

2.2.2. Binocular Cue

The stereo vision system, or in other words, the binocular system uses two cameras to capture the images of the object from two different images of a scene, similar to the human visual system that renders the images for depth perception from the slightly different views of the eyes.

Obtaining the depth information of a scene from a pair of images requires to solve one of the relatively complicated problems in the stereo vision field, called the correspondence problem. Object projections into the planes are represented with respect to two different image coordinates in stereo vision. Hence, the stereo correspondence problem can be described as determining the coordinate difference between the two images of the object. The term stereo matching refers to solving this correspondence problem [19]. The outcomes of stereo matching are usually

represented by a disparity map whose intensity describes the coordinate difference between corresponding image points.

Generally, the stereo vision system has two identical vision cameras, which capture the left and right images of an object to obtain the disparity as Figure 6 shows, while Figure 7(a and b) shows a typical configuration of a stereo vision system. The 3D point P on the surface of a real object is projected into the image planes of the cameras. Therefore, two 2D points, W1 and W2, on the image planes are the projections of point P. The correspondence problem is to find the coordinates of points W1 and W2 between stereo images. The 3D coordinates of the point P can be obtained from the following equations [4]:

$$z = f \cdot \frac{b}{(x_l - x_r)} = f \cdot \frac{b}{d} \tag{3}$$

$$x = x_l \cdot \frac{z}{f} \text{ or } b + x_r \cdot \frac{z}{f} \tag{4}$$

$$y = y_l \cdot \frac{z}{f} \text{ or } y_r \cdot \frac{z}{f} \tag{5}$$

where (x,y,z) are the 3D coordinates of the point P, f is the focal length, the corresponding image points are $(x_l,\ y_l)$ and $(x_r,\ y_r)$, b is the baseline, and d is the disparity of the point P.

As described previously, the typical pipeline for the correspondence problem involves finding the stereo matches. But recently, the convolutional neural networks (CNNs) have been utilised to learn how to find corresponding point. CNNs yields significant gains compared to traditional methods in terms of both speed and accuracy. However, finding accurate corresponding points positions in inherently ill-posed regions such as repeated patterns, occlusion areas, reflective surfaces, and textureless regions is still hard. Consequently, the stereo matching needs to include local support from global context information. Some studies used CNNs to compute patch-wise similarity scores [20][21], with conventional cost aggregation and disparity computation methods. These approaches achieved a state of the art accuracy, but they are limited by the conventional matching cost aggregation step, frequently produced faulty predictions in reflective regions, occluded regions, and around object edges. Other studies seemed to improve the performance of conventional cost aggregation [22][23].

Recently, end to end deep neural network models have become popular. For example, GCNet [24] included the feature extraction, disparity estimation, and matching cost aggregation into a single end to end deep neural network model to get state of the art accuracy on various benchmarks. PSMNet [25] used a pyramid stereo matching network consisting of two main modules that are 3D CNN and spatial pyramid pooling (SPP). The 3D CNN module learns to regularize cost volume using a stacked hourglass block. In contrast, the SPP module takes the advantage of the capacity of global context information by aggregating context in multiple scales and locations to form a cost volume.

(a) (b) (c)

Figure 6. (a) Left image; (b) Right image; (c) The measured disparity from left and right images.

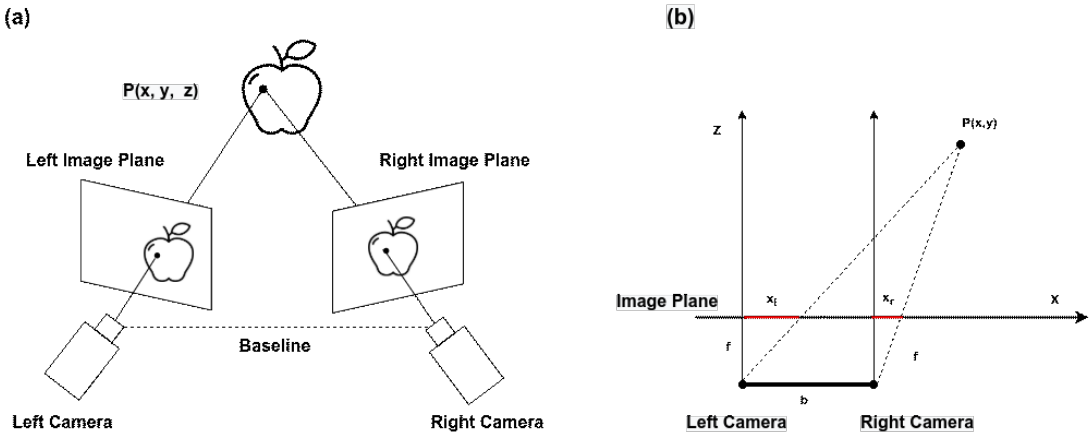

Figure 7. (a) A typical stereo vision system; (b) Simple model of optical axes of two parallel cameras.

2.3. 3D Technology Comparison

In this section, a general comparison of the different 3D imaging and sensing technologies is presented in Table 1. The comparison comprises of three different 3D sensing technologies ranging from stereo vision to structured light and Time of Flight. Moreover, an overview of common 3D sensors is presented in Table 2.

Table 1. Comparison of 3D imaging techniques.

	Stereo Vision	Time of Flight (ToF)	Structured Light
Latency	Medium	Low	Medium
Power Consumption	Low	Medium/High	Medium
Bright Light Performance	Good	Medium	Medium/Weak
Low Light Performance	Weak	Good	Good
Depth Accuracy	mm to cm	mm to cm	mm to cm
Resolution	Camera Dependent	Roadmap to VGA	Camera Dependent
Range	Mid range	Short to long range	short to mid range
Scanning Speed	Medium	Fast	Fast
Active Illumination	No	Yes	Yes

Table 2. Comparison of 3D sensors.

Sensor	Technique	Resolution	Frame rat	Range
ZED camera	Stereo vision	4416x1242	15 fps	0.3–25 m
Bumblebee2	Stereo vision	648x488	48 fps	0.1–20 m
Bumblebee XB3	Stereo vision	1280x960	16 fps	0.1–20 m
DUO3D stereo camera	Stereo vision	640x480	30 fps	—
RealSense R200	Structure light	640x480	60 fps	3–4 m
RealSense R300	Structure light	640x480	60 fps	0.2–1.2 m
RealSense ZR300	Structure light	480x360	60 fps	0.5–2.8 m
Microsoft Kinect1	Structure light	320x240	30 fps	1.2–3.5 m
Microsoft Azure Kinect	TOF	320x288	30 fps	0.5–5.46 m
MESA SR4000	TOF	176x144	54 fps	5/10 m
MESA SR4500	TOF	176x144	30 fps	0.8–9 m
Microsoft Kinect2	TOF	512x484	30 fps	0.5–4.5 m
Argos3D P100	TOF	160x120	160 fps	3 m
Argos3D P330	TOF	352x287	40 fps	0.1–10 m
Sentis3D M520	TOF	160x120	160 fps	0.1–5 m

3. LEARNING-BASED SEMANTIC SEGMENTATION

In this chapter, we present an overview of relevant background knowledge, techniques and algorithms used in semantic segmentation that forms the basis of this research. In the first part, we present the computer vision pipeline, some details about deep learning, CNN architecture, and the most common CNN backbones (inception and ResNet), which are currently being used as backbones for many semantic segmentation models. As well as, we review some deep learning techniques, which are transfer learning and data augmentation. In the second part, we present some details about semantic segmentation and its state-of-the-art networks.

3.1. Computer Vision Pipeline

There are various applications in Computer Vision, but usually, a typical vision system uses a similar process of distinct steps to analyze image data. These are referred as a computer vision pipeline. This pipeline starts by receiving images and data, then processing that data, performing some analysis and recognition, and finally makes a prediction based on the extracted information, as Figure 8 shows.

The input image passes through the classification pipeline as follows: first, a computer receives visual input as an image or a sequence of images forming a video. Each image is then sent into some pre-processing steps in the purpose of standardizing each image. Standard pre-processing steps include resizing an image, rotating, blurring, and converting the image from one colour to another like converting RGB images to grayscale. Only by standardizing each image, for instance making them of the same size, it is possible to further analyze them in the same way. Next, the feature extraction step starts. Features are unique properties and information in the image that are used to classify its objects. For example, some features that distinguish a motorcycle are the shape of the wheel, mudguards, and more. The product of this process is a feature vector that identifies the object.

As a final step, the features vector is fed into a classification model that predicts the class of the image. The classification step is done by either traditional ML algorithms such as SVM and Random Forest, or deep neural network algorithms like CNNs. The traditional ML algorithms might achieve excellent results for some CV problems, but CNNs do better in the classification task. Neural networks automatically extract useful features from the images and act as a classifier at the same time. Thus CNN is what we will discuss in the next section.

3.2. Deep Learning and Neural Networks

Deep learning is a subset of methods in the machine learning toolbox, primarily using artificial neural networks (ANNs), which are a class of learning algorithms loosely inspired by the human brain. DL approaches use many layers of nonlinear processing units for transformations and representations. The input of each layer is the output of the previous layer, and the hierarchical representations can be obtained by different levels of abstraction. These algorithms use neural networks to perform both supervised

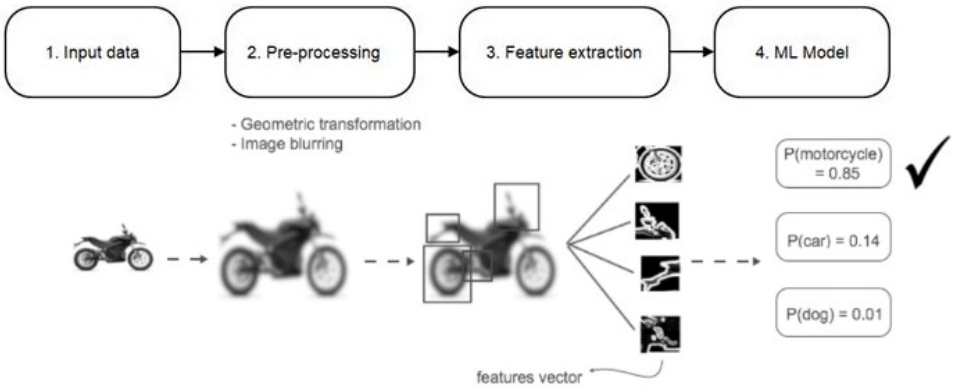

Figure 8. A typical computer vision pipeline with an example.

and unsupervised prediction, and their applications include classification (supervised) and pattern analysis (unsupervised). Nowadays, the accelerated development of deep learning has been achieved by three main reasons that are powerful computation ability, massive data, and innovative algorithms. Many applications of DL are used to solve a wide variety of real-life problems, as well, more and more fields will be facilitated by DL in the future. The basic type of DL approaches are based on neural networks that consist of different layers; each layer contains several units (neurons) that are typically of the form:

$$y = f(wx + b) \tag{6}$$

where y is the activation of each unit that represents a linear combination of input vector x, learnable parameters w, and a basis b, followed by a nonlinear activation function $f(\cdot)$ that can be a sigmoid function or restricted linear unit. Figure 9 shows a typical Neural Network and its building block of a neuron.

The deep neural networks stack multiple layers with different connection structures, which are called as architectures. There are various DL architectures, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), deep belief networks (DBNs), generative adversarial networks (GANs), and stacked autoencoders (SAEs) that have been proposed and successfully used in many domains, and they have achieved state-of-the-art results on many tasks [26][27]. These architectures are considered fundamental ones and can be combined or extended to produce new structures for some specific tasks.

After defining the network architecture, a loss function must be defined, which describes the relationship between the output of the model and the real data. The loss function is used to assess how good the model is at making predictions. The deep neural network learns to map a set of inputs to a set of outputs from training data. The learning process is cast as an optimization problem and an algorithm is used to navigate the space of possible sets of weights the model may use to obtain good or good enough predictions. Typically, a neural network is trained using the gradient descent optimization algorithm and the weights are updated using the backpropagation of the loss function. The gradient descent optimization algorithm seeks to adjust the weights so that the next evaluation reduces the error, meaning that the optimization

algorithm is navigating down the slope of the error function, as Figure 10 shows. The parameter update step of the gradient descent looks as follows:

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w) \tag{7}$$

where the parameter α is the learning rate, w is the parameter updated, and $J(w)$ is the cost function.

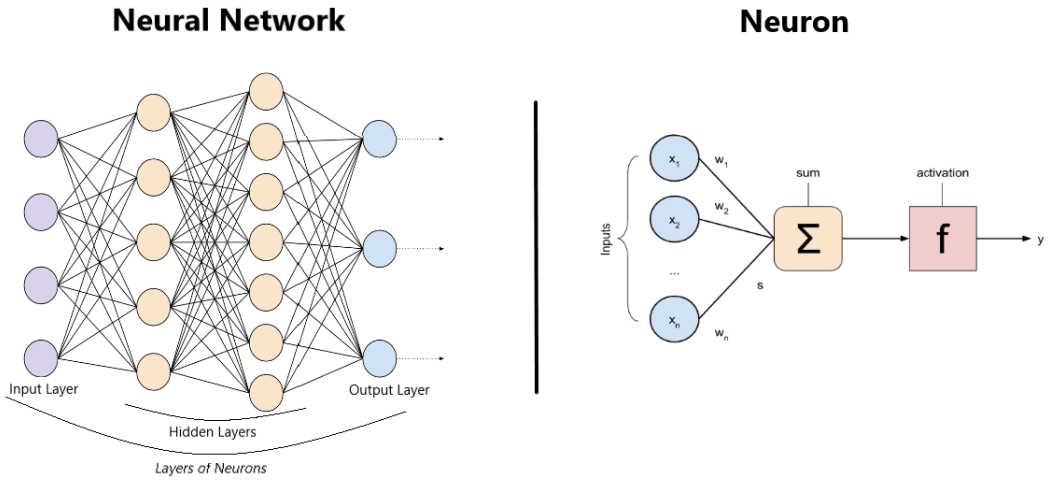

Figure 9. A simple representation of an artificial neuron and Neural Network.

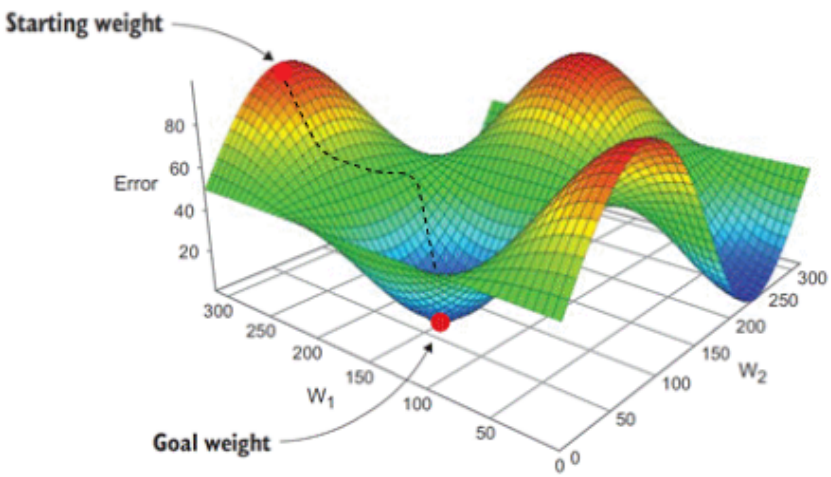

Figure 10. Gradient descent algorithm over the error function.

3.3. Convolutional Neural Networks Basics

Convolutional Neural Networks are the fundamental building blocks of modern computer vision, and the CNNs consist of a sequence of layers, each transforming one volume of activations to another one through linear or non-linear operators. The three main different types of layers to build CNNs are convolutional layer, pooling layer, and fully-connected layer. Figure 11 shows an example of the CNN architecture proposed by Zisserman and Simonyan called VGG16 [28].

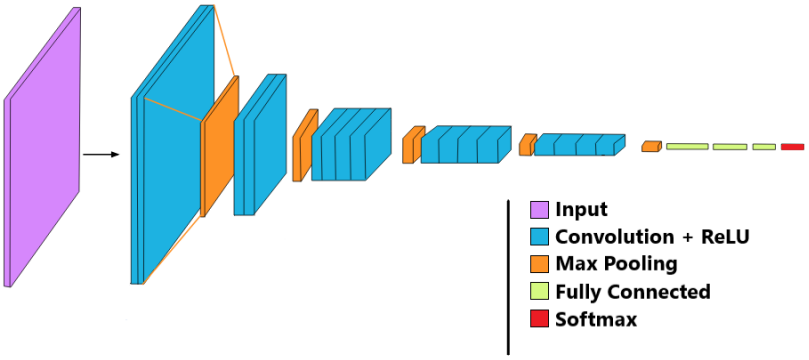

Figure 11. The basic architecture of a CNN (VGG16 Architecture).

3.3.1. Convolutional Layer

The convolutional layer is the fundamental building block of a CNN. It acts like a feature finder window that slides over the image pixel by pixel to extract features that distinguish the objects in the image. The layer's parameters consist of a set of kernels, which have a small receptive field that extends within the full depth of the input size.

At the forward pass, each filter is convolved over the width and height of the input size, calculating the dot product between the entries of the filter and the input producing a two-dimensional activation map of that filter. Therefore, the network learns filters that activate when they see some specific type of feature at some spatial position in the input. Especially, when dealing with high dimensional inputs like images, it is unreasonable to connect neurons to all neurons in the preceding layer because such a network architecture does not consider the spatial structure of the data.

There are three parameters that control the size of the output volume that are the depth, stride and zero padding. First, the depth corresponds to the number of filters that are used. Second, the stride controls how columns around the spatial dimensions are allocated. Third, the size of zero padding allows controlling the spatial size of the output volumes. Figure 12 shows an example of the convolution operation in the convolutional layer, which is applied in a small region of the input data, where the input image dimension is 1x5x5. The number of filters is 1 (K= 1). The convolution filter size is 3 (F= 3). The zero-padding is 1 (P= 1), and the stride is 2 (S= 2).

Hence, by using the equation for calculating how many neurons fit, is given by (W-F + $2P$) /S + 1, the output size has a spatial size $(5 - 3 + 2)/2 + 1 = 3$.

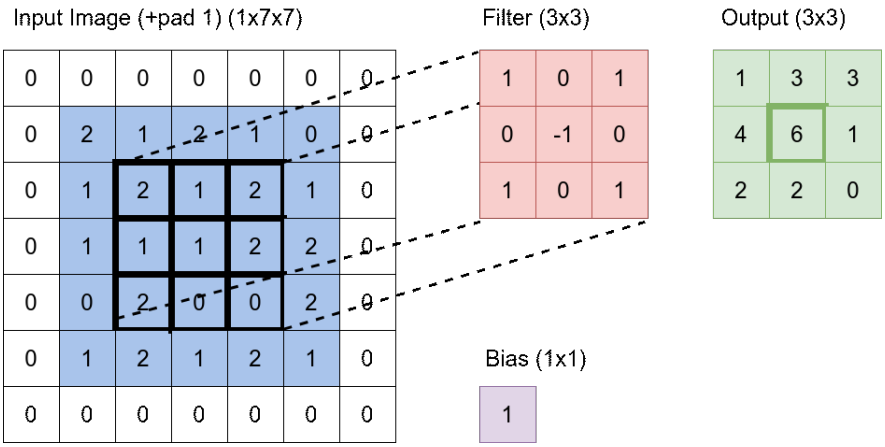

Figure 12. An example of the convolutional layer in a CNN architecture.

3.3.2. Pooling Layer

A different fundamental part of CNN is pooling, which is a form of nonlinear down-sampling. There are different nonlinear functions to implement pooling (POOL) layer. The most common one is Max-pooling, which splits the input image into a set of non-overlapping rectangles and for each such sub-region yields the maximum value. Alike to convolutional kernels, pooling kernels are windows with a specific size that slide over the input image with a stride value. The difference is that they do not have weights. All they do is slide over the feature map generated by the preceding convolutional (CONV) layer and pick the max pixel value to move along to the next layer and ignore the remaining values. The pooling layer operates separately on every depth slice of the input image and resizes it spatially. The typical form is a pooling layer with a filter of size 2x2 used with a stride of 2 that downsamples at every depth slice of the input image.

Another type of pooling layers is average pooling which was often used in the past but has recently fallen out of favour compared to the max-pooling layer, which has been observed to operate better in practice. Because of the excellent dimensionality reduction, the current trend in the research is towards discarding the pooling layer altogether [29] or using smaller filters [30].

3.3.3. Fully-Connected Layer

After passing the image through the feature learning process using the CONV+POOL layers, the high-level reasoning in the CNN is done through fully connected layers (FC layer). The FC layer neurons have full connections to all neurons of the preceding layer. The activations of a FC layer can be calculated using a matrix multiplication plus a bias offset.

In CNNs, FC layers encode the feature volume produced by CONV layers to a long tube of features called a feature vector. Usually, FC layers result in the highest number of parameters in CNNs, which is likely to lead to overfitting. Hence, some recent approaches [31] [32] remove the full connections between the final CONV layer and the following FC layer. Therefore, the total amount of parameters is largely reduced.

3.3.4. Activation Function

Activation functions are proposed to further ensure the nonlinearity of the network [33] because they transform the linear combination of the weighted sum into nonlinear models. There are many types of activation functions. The most common types of them are rectified linear unit (ReLU), Leaky ReLU, Parametric ReLU, Sigmoid, Hyperbolic Tangent Function (tanh), Softsign, and Softmax function. Sigmoid function is commonly used in binary classification. As pointed out in [34], using Sigmoid function on the back-propagated gradients can be easily saturated, which make it difficult for updating the weights of the network. On the other hand, ReLU preserves information about relative intensities as information passes through many layers of feature detectors [35]. Leaky ReLU [36] is a variant of ReLU to fix the dying ReLU problem by having a small negative slope.. Later on, to overcome the limitations of leaky ReLU and ReLU, PReLU [37] has demonstrated a new method in learning the slopes of the negative part from data rather than predefined.

3.3.5. Weight Initialization

There is no unique method of initializing the network weights. Glorot et al. [34] proposed the Xavier initialization as a new weight initialization method such that the variance of the backpropagated gradients is roughly constant across layers. Whereas the gradients have initially approximately the same magnitude, they diverge from each other as the training progresses. Thus, this is one of the advantages of this method, since having gradients of different magnitudes at different layers may yield to a slower training process. The variances of gradients are used as essential criteria in measuring the learning step during the training process. He et al. [37] introduced MSRA/He Initialization as another weight initialization regarding PReLU. The motivation for proposing this method is by considering the asymmetric distribution of the activation function.

3.3.6. Regularization Solutions

Regularization is a way to prevent the overfitting problem. The purpose of regularization is to modify the learning algorithm to make the model perform well. One of the most widely used regularization solutions to the overfitting problem is known as Dropout, that was proposed by Srivastava et al. [38]. The idea of Dropout is to train a group of neural networks and average the results rather than training only a single

neural network. Dropout builds new neural networks, by dropping out units with a predefined dropping out rate.

Ioffe and Szegedy [39] proposed the batch normalization, which is another regularization technique. Batch normalization is a good method to speed up and increase the stability of the training process by decreasing the oscillation of the loss function. Batch normalization normalizes the layer's output by subtracting the batch mean and dividing it by the batch standard deviation. After that, Batch normalization adds the standard deviation parameter γ and the mean parameter β.

3.4. Convolutional Neural Networks Backbones

3.4.1. Inception

The inception network was invented in 2014, when Szegedy et al. [40] proposed a 22-layers deep network using inception modules, as shown in Figure 13. This network is called GoogLeNet. The idea of the inception module is to act as a multi-level feature extractor by using 1x1, 3x3, and 5x5 CONV with the same module of the network. The output of these filters (feature extractor) is stacked along the channel dimension and before being fed into the next layer in the CNN. The overall number of weights is smaller than both in VGG and ResNet, and they are further reduced in the work inspired by inception, called Xception [41]. GoogLeNet is deeper than VGGNet while reducing the number of parameters by 12 times, from around 138 million to 13 million parameters, and achieving significantly more accurate results.

3.4.2. ResNet

Residual Neural Network was developed in 2015 by Kaiming He et al.[42], in order to avoid the problem of vanishing and exploding gradients [34], and to allow to train very deep neural networks with 50, 101, and 152 layers while still having lower complexity than smaller networks like VGGNet of 19 layers. As shown in [42], when the CNN depth increases, accuracy gets saturated and then degrades rapidly, which hampers convergence from the beginning. To overcome this problem, a novel architecture with shortcut connections called residual module was proposed, where these shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers, as shown in Figure 13. The ResNet features heavy batch normalization for the hidden layers. Moreover, ResNet was able to achieve a top-5 error rate of 3.57% in ILSVRC15 [43], which beats the performance of all prior CNNs.

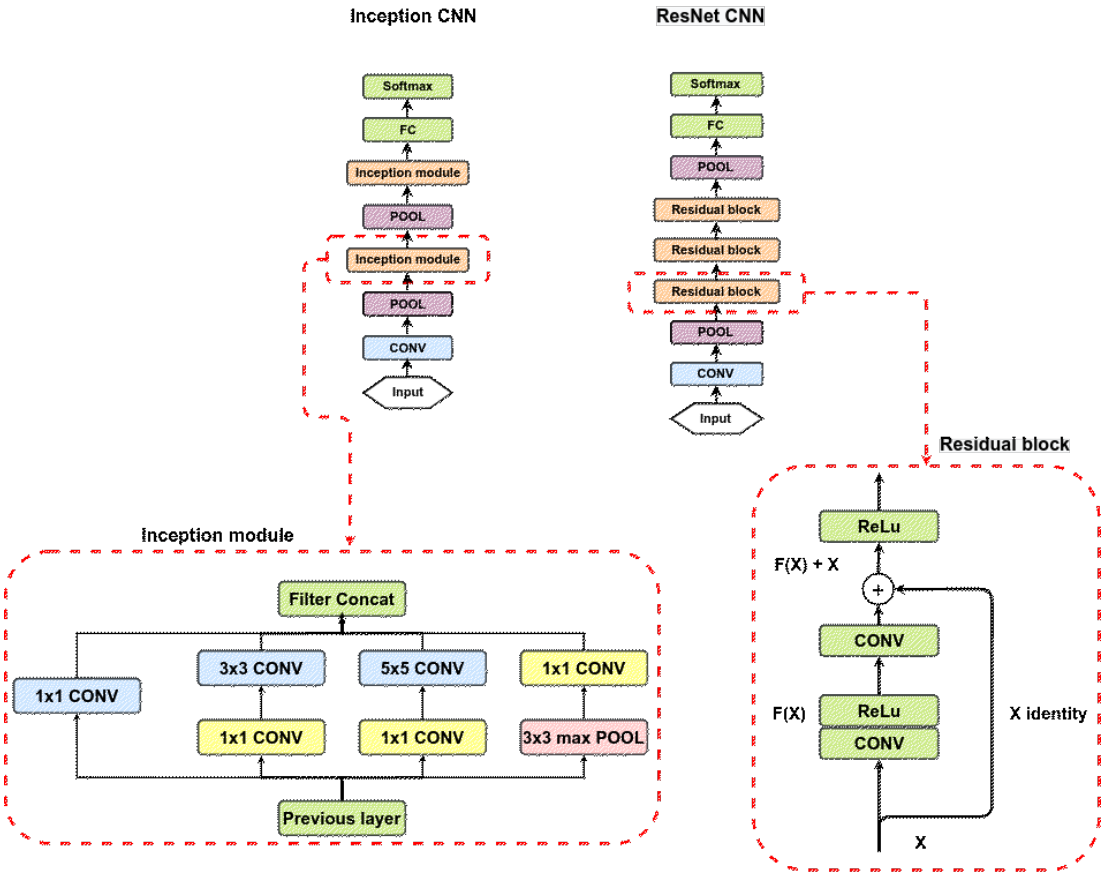

Figure 13. The inception module and the Residual learning block.

3.5. Deep Learning Techniques

3.5.1. Transfer Learning

Training an entire CNN from scratch usually requires a massive dataset such as ImageNet, and computing power. Since it is time-consuming and expensive work that has already been done, it is a bright idea to use parts of the trained model to solve the classification problem. Actually, it is possible to transfer what the CNN has learned from one dataset to a new one, through transferring the knowledge. Transfer learning (TL) is the method of learning a new task by relying on an earlier learned task. The learning process can be much faster, more accurate, and needs less training data.

The transfer learning idea is smart, and it is one of the most important techniques of deep learning as it can be successfully applied when using CNNs. TL uses a neural network that was already built and trained on a large dataset in a certain domain and using this pre-trained CNN as a starting point to train the CNN on a new task. Transferring the knowledge of a network that has been previously trained (pre-trained model) to a new one requires us to remove the classifier part of the network and freeze the feature extraction part. After that, we attach the new classification layers to the network. In this way, we can train a CNN by reusing the knowledge learned on a massive dataset and embedding it into the model. Therefore, transfer learning leads to

two significant advantages, which are speeding up the training process, and potentially mitigating the overfitting problem.

3.5.2. Data Augmentation

Data augmentation is a simple way to extend the size of the dataset. It is done by applying a set of transformations on the training data. It aims to avoid the overfitting problem, and to make the model aware that certain input variations are possible. Therefore, it improves the performance of the model on a variety of input data. There are many image augmentation techniques that can be used. To mention a few flipping, rotation, zooming, scaling, lighting condition, and many other techniques can be applied to a dataset to make the network learn with a variety of training images. However, before applying any of these techniques to a dataset, we have to be sure that those transformation techniques are meaningful for data type, and the task at hand.

3.6. Semantic Segmentation

Object recognition is a fundamental task in computer vision. Humans have a remarkable ability to analyse an image and separate all the components present in an image, thoroughly parsing it and easily recognising and localising the objects. However, building machines that can achieve the object recognition effortlessly is particularly challenging. Object recognition can be divided into three sub-problems from easy to hard based on the level of complexity as follows: The first one is the image classification task which is identifying objects within an image and providing image-level labels. The output labels are independent of object locations. The second one is the object detection task which is identifying objects within an image and creating a bounding box surrounding each detected object. The third one is the semantic segmentation task which is identifying objects within an image and providing a label of a known semantic class to each pixel present in that image.

Among these three tasks, semantic segmentation is the most challenging and critical task in object recognition, and it covers the way towards complete scene understanding. Many essential applications are provided from the task of semantic segmentation, such as medical imaging [44][45], autonomous driving [46][47], and human-machine interaction [48]. Image segmentation is the most widely studied task by far in computer vision, where each pixel is annotated using the identifier of a particular object to generate a segmentation map, so that each pixel is assigned a class label. Image segmentation sorts pixels into larger segments, eliminating the need to consider single pixels as units of observation.

Most of the recent semantic segmentation works are focused on recognizing and partitioning image into meaningful classes of objects like for example, person, dog, cat in the PASCALVOC 2012 challenge [49]. Some classes have specific shapes, sizes, and identifiable parts. For example, a person has a head, two legs and two arms. In recent works, more attention has been given to the stuff classes, such as water, cloud, and sky but also another side of classes based on material types such as skin, plastic, and wood which are amorphous and have no distinct parts [50][51][47]. Recently,

CNNs are widely used for semantic segmentation. They achieved state-of-the-art performance, for instance, using the fully convolutional network [52][53][54][55]. However, the current class segmentation approach still suffers from the common problems of global context embedding, image quality variations, and imbalanced training set [56][57].

3.6.1. Related Work

In this section, we review advances and related works in semantic segmentation, where encoder-decoder deep learning methods have become the standard approach for semantic segmentation problems.

Earlier studies rely on pre or post-processing and encode segmentation relations using Conditional Random Fields (CRFs) [58][59][60]. Maire et al. [58] presented an alternative framework for people detection and segmentation, in which it integrates the outputs of a top-down part-based person detector in a generalized eigen problem, producing pixel groupings. Plath et al. [59] introduced an approach that joins local image features with a CRF and an image classification approach to combine global image classification with local segmentation. A different category of CRFs called Hierarchical Conditional Random Fields (HCRF) has been introduced in [60].

The era of using fully convolutional networks (FCNs) started with [61] and [62], where FCNs were applied to detection [61] and semantic segmentation [62]. Sermanet et al. [61] proposed a CNN sliding window approach which recognizes, detects and locates objects. All these mentioned methods while being effective to their tasks still were restricted to be patch-based, which can be inefficient for massive inputs data. Pinheiro and Collobert [62] used a recurrent neural network (RNN) for scene segmentation, which also works on image patches.

On the other hand, the model proposed by Long et al. [54] allows training the network end-to-end for semantic segmentation tasks using the entire image as input. The proposed approach does not need any pre or post-processing method compared to earlier approaches. The concept of this approach presents a state-of-the-art performance in generic semantic segmentation tasks. It replaces the FC layers of a CNN by convolutional layers that produce coarse score maps. As well as, another FCN architectures like [63][64][65] were important for semantic segmentation. Liu et al. [63] presented the ParseNet, which focuses on global pooling and can model global context information directly. Ronneberger et al. [64] proposed a new architecture called U-Net, which is an up-convolutional architecture for image Segmentation of microscopy images. Badrinarayanan et al. [65] proposed a different FCN architecture called SegNet which focuses on computational efficiency. Their main contribution is enhancing the performance of the network by the use of pooling indices computed in the max-pooling step at the decoder.

Another group of semantic segmentation research is more focused on scene understanding for the autonomous vehicle. For such scenarios, the drivable area needs to be obtained. The state-of-the-art benchmarks of autonomous vehicle segmentation are deep learning approaches such as [66][67]. Other works on semantic segmentation were designed to incorporate context explicitly like Deeplab-V2 [68] and ParseNet [69]. DeeplabV2 proposes Atrous Spatial Pyramid Pooling (ASPP), which includes

several parallel atrous convolutions with filters at different sampling rates and effective fields-of-views, while, ParseNet involves global pooling features to explicitly add context information.

3.7. Semantic Segmentation Networks

Currently, the most successful state-of-the-art semantic segmentation tasks are driven by the recent progress in classification with CNNs, pixel-level prediction achieved great success inspired by the FCN approach. The insight of this approach is to transform the existing and well-known classification networks like VGG, Inception, and ResNet, into FCN by replacing the FC layers with the convolutional layers to output the spatial maps instead of classification scores. Those spatial maps are later upsampled to create dense per-pixel labelled outputs. In general terms, the FCN-based semantic segmentation models consist of two main parts that are the encoder and the decoder. The encoder usually is the part of the classification network like ResNet with its FC layers removed. The encoder produces feature maps, and then decoder upsamples those maps to pixel-wise predictions.

In the next subsections, we will review some of the state-of-the-art FCN-based (encoder-decoder) semantic segmentation networks.

3.7.1. DeepLab

DeepLab is a state-of-art and one of the most successful deep learning model for semantic segmentation designed by Google, where the aim is to assign semantic labels (for example, person, cat, dog) to every pixel in the input image. The current implementation of DeepLab includes the following features: From DeepLabv1 [70], they use atrous convolution to explicitly control the resolution at which feature responses are computed within Deep CNN. From DeepLabv2 [68], they use atrous spatial pyramid pooling module (ASPP) to robustly segment objects at different scales with filters at different sampling rates and effective fields-of-views. Figure 14 shows the ASPP module. From DeepLabv3 [52], they include batch normalization parameters to facilitate the training. Besides, they augment the ASPP module with the image-level feature to capture longer range information. Especially, they apply atrous convolution to extract output features at different output strides during training and evaluation, which efficiently allows training batch normalization (BN) at output stride equal 16 (OS =16) and achieves high performance at output stride equal 8 (OS =8) during evaluation. From DeepLabv3+ [53], they extend DeepLabv3 model to include a simple yet effective decoder to improve the segmentation results, especially along object boundaries.

3.7.2. SegNet

In 2017, Badrinarayanan et al. [65] proposed a new FCN architecture called SegNet for semantic segmentation. The encoder of this architecture is based on VGG-16, and

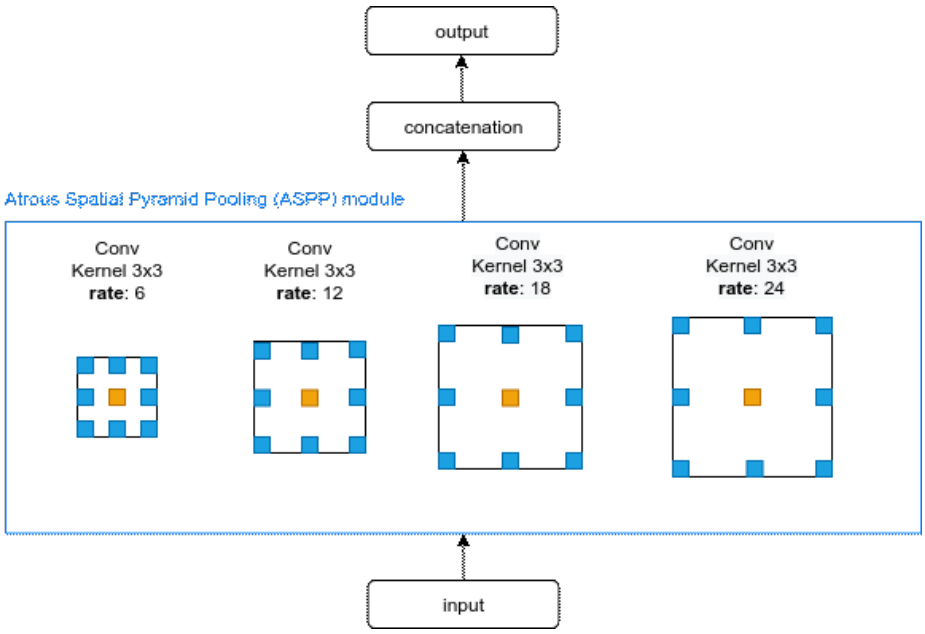

Figure 14. Atrous Spatial Pyramid Pooling (ASPP) module.

its decoder is formed by a set of upsampling and convolutional layers which are at last followed by a softmax classifier to predict pixel-wise labels. The novelty of SegNet lies in the way in which the decoder upsamples its lower resolution input feature maps, where the decoder uses pooling indices calculated in the max-pooling step of the corresponding feature maps in the encoder phase to perform non-linear upsampling. At the final step, when the decoder network maps the low-resolution encoder feature maps to the original input resolution, they are fed to the softmax classifier to produce the dense feature maps.

3.7.3. GCN

Peng et al. [71] introduced Global Convolutional Network (GCN) for semantic segmentation. They followed two design principles. The first principle is: From the localization design, where the network structure should be fully convolutional to retain the localization performance without using FC or global pooling layers because these layers will discard the localization information. The second principle is: From the classification design, where large kernel size should be used in the network to allow densely connections between feature maps and dense per-pixel classifier, which improves the capability to handle various transformations.

3.8. Material Recognition and Segmentation

Material recognition is an inherently challenging problem in computer vision, primarily because of the wide variation in form between different instances of a given material and between different materials. But recently, there has been significant progress in terms of accuracy on benchmark datasets. Most existed methods proposed treat material recognition as object recognition with different classes. Those methods often use large image patches that cover parts or whole objects and scenes similar to performing object recognition, which combines visual cues of materials and image context.

Material recognition is different from object recognition. Adelson [72] mentioned this distinction in his study of THINGS vs STUFF. The fundamental difference between them highlights the critical difference between materials and objects. STUFF refers to materials, and THINGS refers to objects. Materials (STUFF) may not necessarily be recognized by having a special shape. For example, a cup is an object with a typical cylindrical shape. It is often made of ceramic. The cup as an object can be used as a cue to recognize the material as ceramic, and likewise, the ceramic can be used as a cue to recognize that an object might be a cup. However, not all ceramic THINGS are cups and relying on shape cues to recognize ceramic, like earlier methods do is a limited approach.

Most of the existing works on material recognition usually make the recognition at an image patch level. They use a significant part of the scene, sometimes even the whole image. Sharan et al. [73] proposed the earliest design of such classification with the Flickr Materials Database (FMD). FMD use the entire image as the image patch, and each image on the dataset contains a single primary material of interest similar to image classification dataset. Recently, Bell et al. [55] demonstrated per-pixel material classification and segmentation using large-scale annotated training images, the Materials in context (MINC) dataset. They used an architecture based on a combination of CNN and CRF networks for classification. They used a large image patch for each pixel. The patch is about a quarter of the whole image, which mix in the object or location context to the material nature.

Material recognition would require a large training dataset that contains the product space of objects and materials. Zhang et al. [74] have recently presented an impressive performance on the FMD, but their model concentrated only on single patch predictions. Wang et al. [75] also introduced a model for accurate dense per-pixel material recognition using 4D light field images. These methods combine materials and context of the image throughout the recognition pipeline. Therefore, the use of context as a property to decrease ambiguity seems promising. On an unrelated study, Iizuka et al. [76] use scene place predictions to improve the accuracy of the greyscale image.

4. DATASET GENERATION

4.1. Data Collection

Data plays a significant role in all major advancements of computer vision where annotated benchmark datasets serve many goals such as providing training data, evaluating progress, and helping the community to focus the efforts on the next steps towards developing visual intelligence. We rely on access to large collections of annotated images that cover the variability of the visual world. Collecting this large-scale labelled data is challenging and expensive, demanding the development of new techniques for data collection and annotation. In the collecting process, the important question is what annotations should be collected, and answering this question needs to make decisions about the type of images and annotations, and the scale of annotation. Different types of data come with different associated costs, including formulating the desired dataset, developing the annotation procedure and user interface design, and annotator time.

The task of semantic image classification is a fundamental task in semantic image understanding. There are two types of segmentation annotations: semantic segmentation and instance segmentation. Semantic segmentation datasets provide an outline around contiguous areas that share a similar semantic property while instance segmentation datasets provide an outline of every instance of the objects.

Some examples of instance segmentation datasets include PASCAL VOC [77], LabelMe [78], and MS-COCO [79]. PASCAL VOC has 20 classes in around 7 thousand images. LabelMe was created by recruiting volunteers to draw polygons around object instances. Another dataset called Semantic Boundaries Dataset [80] extends the annotations of the PASCAL VOC by five times. The COCO dataset is one of the largest instance segmentation datasets with around 2.5 million object instances manually segmented within more than 328,000 images. Semantic segmentation datasets provide semantic labels for every pixel in the image. Some examples of semantic segmentation datasets include SIFT Flow [81] and PASCAL-context dataset [82]. SIFT Flow is a dataset of 2,688 images and 33 classes labelled using the LabelMe annotation tool. The PASCAL-context dataset annotates the original PASCAL VOC images with 520 new classes.

Some works add more detail to the segmentations like the works that was introduced by Bell et al. [83][55], that segment and annotate material characteristics. OpenSurfaces [83] has more than 22 thousand images accurately labelled, and it provides named objects, named materials, rectified textures, and other characteristics. Materials in Context Database [55] has around three million material images that were annotated by using the three-stage Mechanical Turk pipeline. These datasets help achieving a deeper understanding of the pixel-level segmentation.

Collecting segmentation datasets is very time-consuming and particularly expensive. However, such detailed annotations facilitate the development of computer vision algorithms that can understand the image at a more accurate level.

4.2. Data Annotation

In image segmentation, well-annotated image and video ground truth are necessary for performance evaluation, and those labelled images are used in supervised learning. Accurate and fast image annotation is a well-known problem in computer vision. Image annotation tools look for maximize labelling accuracy and minimizing human effort and time. The existing image annotation tools can be divided into three classes: bounding box-based labelling, boundary-based labelling, pixel-level labelling.

Bounding box labelling is usually used in object recognition and tracking. It is easy to implement because each simple bounding box is defined by two corners. Vondrick et al. developed a crowdsourcing video annotation tool called VATIC, which proposes inter-frame interpolation to create bounding boxes semiautomatically [84]. Doermann and Mihalcik developed a video annotation called ViPER, which allows users to do annotation frame by frame [85].

On the other hand, pixel-wise labelling provides detailed shape descriptions of target objects such as graph cuts, and segmentation, which have been proposed to reduce the need for user intervention. The pixel-wise labelling performs well when the annotated images have relatively flat backgrounds and remarkable foreground. Otherwise, the generated masks are often inaccurate and noisy.

Closed boundaries (boundary-based labelling) are used in most of the image annotation tools, and polygons usually approximate them. The annotation accuracy depends on two factors that are the number of control points and their localization error. Therefore more control points provide more accurate closed boundaries, and for the second factor, the labellers should localize each point very accurately.

There are several state-of-the-art of image annotation tools. Some of them are presented next. LabelMe [78] is a free online annotation tool for computer vision purposes, developed by researchers from MIT Laboratory. It has two types of markers which are polygons and masks. Its annotation result can be exported to XML format. VGG Image Annotator (VIA) [86] is an open-source and offline web-based annotation tool developed by researchers from Visual Geometry Group, University of Oxford. The main advantage of this tool is that it has more marker types, which are dot, line, rectangle, ellipse, circle, polygon, and polyline, while its disadvantage is that it has only one annotation colour for all objects. Its annotation result can be exported only as a CSV file. Labelbox [87] is a commercial online annotation tool for computer vision tasks. It has many types of marker, which are a dot, line, brush, and superpixel. It has the best user experience so far and makes the annotation easier when a polygon marker is drawn on the object, the marker will move near the object border. The annotation result of this tool can be exported into different formats such as CSV, JSON, and COCO.

4.3. Data Analysis and Visualization

Data analysis is a task that tries to discover meaningful information from a dataset. It includes a huge range of activities, from looking for trends and implementing statistical analyses, to analysing different types of data to extract thematic relationships.

Data visualisation refers to any task of presenting information so that it can be described visually. The data visualisation process is to put data into things like charts or diagrams, graphs, animations, or infographics. It helps us to recognise trends, patterns and relationships to extract meaningful information from a dataset. There are many types of visualisations to choose for displaying data, and there are many tools that can be used to create visualisations.

4.4. Dimensionality Reduction

Many machine learning problems include a large number of features, which lead to produce many problems. The most well-known problems are: They make the training very slow and hard to find a good solution. The dimensionality reduction is the process of decreasing the large number of features to the most relevant ones in simple terms. Reducing the dimensionality leads to losing some information and make the system operate slightly worse as most compressing processes it comes with some disadvantages. However, reducing the dimensionality makes the training faster and can filter out some of the noise and some of the unnecessary information.

Most dimensionality reduction applications are used for data visualization, data compression, and data classification. Data Visualization is one of the essential aspects of dimensionality reduction. We can visualize the data on a 2D or 3D plot by dropping down the dimensionality to two or three, and the critical information can be obtained by cluster analysis. There are many techniques that can be used for dimensionality reduction, such as PCA and TSNE. There are two main approaches for dimensionality reduction that are projection and manifold learning. Projection is an approach that deals with projecting every data point of a high dimension, onto a subspace suitable lower-dimensional space. Manifold learning is an approach for dimensionality reduction to non-linear structure data where algorithms for this approach are based on the idea that the dimensionality of many data points is only artificially high.

4.4.1. Principal Component Analysis

Principal Component Analysis or PCA is a linear feature extraction technique and one of the best known dimensionality reduction technique. It works by identifying the hyperplane which is located in close to the data, and then projecting the data into that hyperplane while retaining most of the variance of the original data. The axis that explains the maximum value of variance in the data set is called the principal components (PC1). The axis orthogonal to the PC1 axis is called the second principal component PC2. PCA would find a third component orthogonal to the other two components (PC1 and PC2) if we go for higher dimensions and so on. The visualization of this method always sticks to 2 or 3 principal components, as shown in Figure 15.

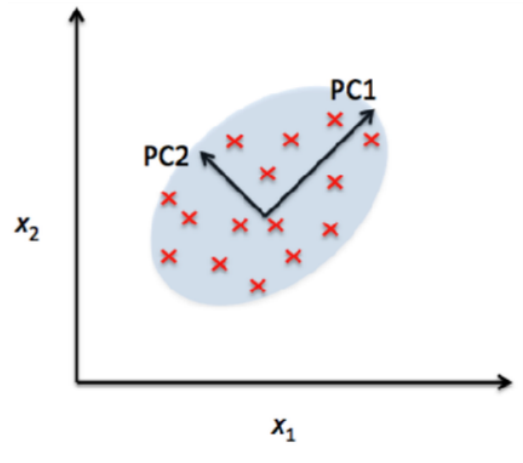

Figure 15. Graphical representation of PCA transformation in two dimensions.

4.4.2. T-Distributed Stochastic Neighbor Embedding

The T-distributed stochastic neighbour embedding or TSNE is a non-linear technique for dimensionality reduction created by Maaten and Hinton. [88] for the visualization of high-dimensional datasets. TSNE brings high-dimensional datasets and degrades its dimensionality to a low-dimensional graph that holds important original information. Each data point has a location in a 2D or 3D map. Thus this technique forms clusters in the data. TSNE decreases dimensionality while trying to keep similar patterns of the data close and dissimilar patterns apart.

4.4.3. TSNE versus PCA

Although both PCA and TSNE have their own pros and cons, some critical differences between PCA and TSNE are: PCA is a mathematical method, but TSNE is a probabilistic one. TSNE is computationally expensive and can take many hours on large datasets while PCA will finish in seconds or minutes. PCA is a linear dimensionality reduction algorithm that focuses on placing dissimilar data points apart. However, in order to represent high dimensional data on low dimension using a non-linear manifold, it is important that similar data points must be kept close together, which is something TSNE does but not PCA. Since PCA is a linear algorithm, it will not be able to describe the complicated polynomial relationship between features while TSNE does. Sometimes different runs of TSNE with the same hyperparameters can provide different results. Hence various plots must be observed before making any evaluation with TSNE, while this does not happen with PCA.

5. EXPERIMENTAL EVALUATION

In this chapter, we will present the details of the implementation and evaluation process of the material segmentation that is based on beam profile analysis technology, that was introduced and discussed over the course of this thesis. The segmentation process pipeline, shown in Figure 16, reaches from dataset generation to the DL architecture and finally to the quantitative and qualitative results.

The segmentation process pipeline shown in Figure 16 is a sequential process of problem-solving using supervised CNN. Collecting data is the first step to be carried out after understanding the nature of the problem. Generally, CNNs need a massive amount of data to reach its best performance. The next step is data preparation and validation. This step should not be ignored as this gives a significant improvement to the performance. This step also tells us that the need for CNNs to be powerful does not only depend on a massive amount of data but also the valid one. Then, building the model and refining its algorithm is the next step after the data preparation step. An effective CNN model is built for the semantic segmentation problem, where improving the model architecture can significantly improve the final results. Next step is the training step, we show in this section the details of the training process. The speed of the training process takes a primary concern. The training speed depends on the computational capabilities (GPU, CPU, and RAM...) and the model architecture. The last step is testing or evaluation of the results and we present in this section the final results of the proposed models. The results are both quantitative and qualitative, where the quantitative results are based on some evaluation metrics, while the qualitative results are visually compared to the ground truth images.

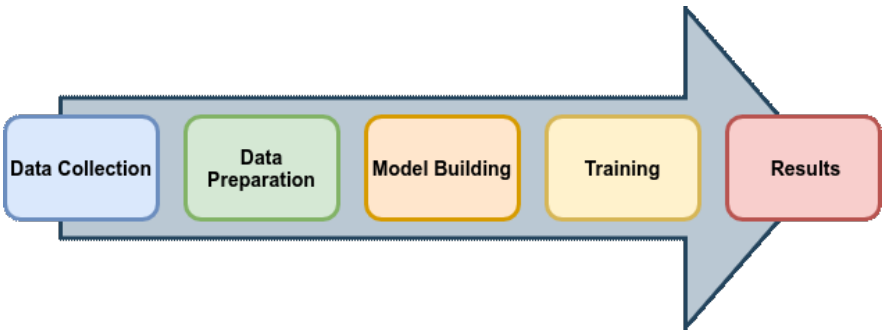

Figure 16. The pipeline of our experiment.

5.1. Implementation Environment

Our Beam Profile Analysis (BPA) sensing solution differs from the conventional technologies. It uses a grid of laser beams projected to the scene and a CMOS sensor for observing the reflected beams. The grid has 2005 spots in a rectangular shape of 40x50 points. The sensor produces image pairs consisting of greyscale images with and without the laser spots (greyscale and laser images) and at full CMOS resolution of 1440x1080. Analysis of the beam profile provides 3D coordinates (x, y, z) as well as feature vectors associated with the points that describe the scene of the frame. These

feature vectors (metadata) are represented as a data point structure, which contains many features saved on a JSON file. Also, these feature vectors have been successfully used to derive statistical models, which allows classifying materials based on specific scattering/reflection properties.

We utilize a Windows 10 Pro (x64) LEGION Lenovo laptop with Intel Core i7 @2.6 GHz, 32 GB RAM, and an NVIDIA GeForce RTX 2070 GPU card for developing and testing the system. The experiments carried out in this work were built under TensorFlow in Python. Most of our development is done in Anaconda distribution of Python version 3.7. Python-based open-source tools: Pandas, Numpy, OpenCV, Scikit-learn, TensorFlow, PyQt and other libraries are used to develop the algorithms. Besides, we used the Visual Studio Code as our IDE and Jupyter notebook to run Python scripts that use the mentioned libraries.

5.2. Data Collection

Training a deep learning algorithm to classify objects and materials requires annotated training and test data. Because the aim in this project is to perform pixel-level material and semantic segmentation, it is necessary to collect a dataset to the material of interest where the images are the grayscale and laser images are obtained with a beam profile analysis (BPA) sensing device, and where those images have been manually annotated and labelled.

In general terms, this section discuss about dataset generation, starting from analyzing the features of the laser spots to choosing the classes that the CNNs can learn from and distinguish between them. Then, we discuss about images collection process, image annotation process, and lastly discuss about dividing and organizing the final dataset.

5.2.1. Data Analysis

The data analysis and the study of the features is a starting point for us to create our dataset and learn the characteristics of our laser spot images on different materials. The data analysis in our case include cluster analysis and classification. Because currently, we lack of knowledge and understanding what are the important characteristics as well as limitations to discriminate between different materials, it basically shows what kind of materials we can use in our dataset and what kind of accuracy to expect at the final stage. In addition, the reasoning behind the study of the features comes from the difficulty to know which materials to include in our dataset. So we have to take this into account when selecting classes because as we mentioned previously in Chapter 3, that material recognition is an inherently challenging problem in computer vision, primarily because of the wide variation in form between different instances of a given material and between different materials. As well as, material recognition is different from object recognition, like what we mentioned before as the difference between THINGS and STUFF. Materials (STUFF) may not necessarily be recognized by having a special shape.

As we know that our BPA sensing technology provides us feature vectors (metadata) associated with the points that describe the scene of the taken frame. This metadata is a data point structure, which contains several features, which are saved on a JSON file. Some of the features describe the information extracted from the grayscale images and other information else. Our data analysis focuses only on the features (information), which are extracted from the grayscale and laser images. Those features are briefly described as follows:

- Colour [GBR]:is the average colour values of the spot's pixels. This feature has three dimensions because it is a vector of 3 colour values, which are green, blue, and red (3 dimensions).
- Brightness:is the normalized sum of pixel values inside a circular region of a spot (1 dimension).
- Background:is the average brightness of the background pixels around the spot (1 dimension).

Choosing the suitable material classes for our dataset is a bit challenging because we should carefully check that the chosen materials are distinguishable between each other, and there is no significant similarity among classes. In our study case, we find that materials like glass and metal (except aluminum) are probably not very good materials to detect, because they have very high specular reflection and tiny diffuse reflection. So we do not consider reflective materials like metal (except aluminum) and glass in our material detection. Some of the classes are very wide and diverse, which will make it difficult to recognise them as the same class. For instance "plastic" or "food" could be anything with very different reflective properties.

Our strategy is to narrow the proposed classes down to more fine-grained categories that are expected to have low intra-class variance and high inter-class variance. For example, we could choose "cotton" that appears to have a similar texture and reflective properties, or choosing just one type of fruit (banana, orange, etc.). So, in general terms, we aim to reduce intra-class variance as much as possible and limit our experiments to a few very distinct materials. After studying and testing different types of materials carefully with the help of cluster analysis and material classification, we ended up choosing 10 classes that are shown in Table 3.

Table 3. The material classes that have been chosen.

Material class	Aluminum	Apple	Broadcard	Cotton	Orange
Label	0	1	2	3	4
Material class	Banana	Potato	Skin	Wood	Paper
Label	5	6	7	8	9

In this study case, we have around 14500 data points, which have been taken from different images and classes. Each data point related to its corresponding spot was labelled individually to its material class. Also, each data point (feature vector) used has the features: brightness, background, colour B, colour G, and colour R.

Cluster Analysis:

The data analysis environment is set up and conventional cluster analysis is performed with the laser spot feature vectors (metadata) to test how different materials included in the dataset can be separated in the feature space. Each laser spot data point has 5 dimensions in total (Colour [GBR], Brightness, and Background). Therefore, we utilise PCA and TSNE as our dimensionality reduction techniques to reduce our data points from 5D to 2D representation and visualise it in 2D plots as Figure 17 and 18 show. Also, we utilise PCA and TSNE to find out if the feature representation can be compressed without sacrificing the classification performance. The results of this cluster analysis, as shown in Figure 17 and 18, are primarily used to choose distinguishable materials and improve the data collection process.

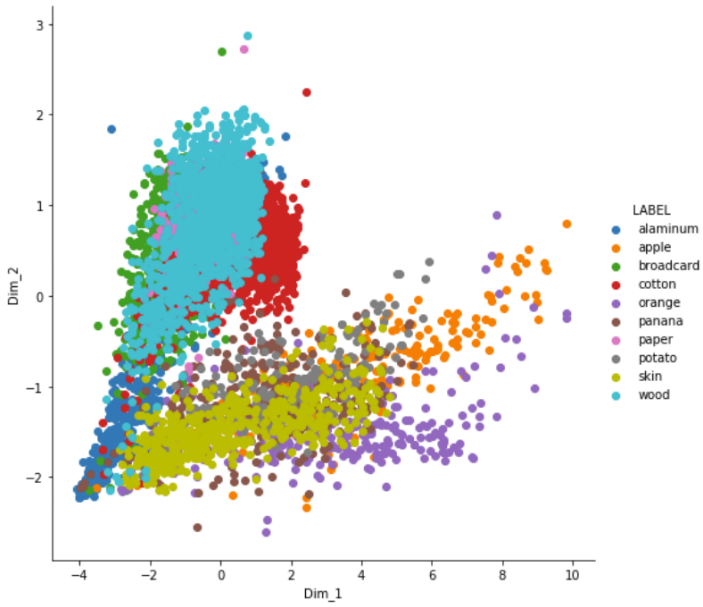

Figure 17. The results of the PCA analysis.

Data Classification:

In the classification process the aim is to classify each data point to one of the material labels that have been chosen before (banana, cotton, wood...) using six different kinds of classifiers that are SVM, Random Forest (RF), Decision Tree (DT), Naive Bayes (NB), K Nearest Neighbors (KNN), and Multi-layer Perception classifier (MLP).

The data points have been taken from different images and classes. We have around 14500 data point that were divided into a training set containing 13000 data points and 1500 data points were included to a testing set. The training and testing samples were taken from all the classes. The results of the data point classification are shown in Table 4.

Discussion:

Depending on the cluster analysis of PCA and TSNE from Figure 17 and 18, we can see that the 10 selected material classes provide us good distinct clusters like

Figure 18. The results of the TSNE analysis.

Table 4. The results of the data points classification.

RF Classifier Accuracy	MLP Classifier Accuracy	SVM Classifier Accuracy
45.69%	53.54%	48.27%
DT Classifier Accuracy	KNN Classifier Accuracy	NB Classifier Accuracy
48.06%	43.76%	42.58%

between skin and cotton or between aluminium and apple. However, some materials appear overlapped in the feature space like skin, apple, orange, banana, and potato. On the other hand,the classification results from Table 4 demonstrate a satisfactory classification accuracy from all the classes ranging from 40% to 55%, which supports our cluster analysis observations. We should also take into consideration that there are some outliers and wrong manually labelled data points because of the difficulty of the labelling process and camera limitations. Generally, the data analysis shows us promising results about the distinguishability of the chosen materials, which makes our dataset useful, and let the CNNs learn from it pretty well and give us good accuracy at the end.

5.2.2. Collecting Process

Performing all stages of data collection, annotation, and preparation takes a lot of time and effort, because, the validity is the main concern of those stages and the number of data samples has an important role as well. Those stages have to be performed by a manual operation. They are the hardest parts of this work and their result is crucial to the result of the CNN model.

As we know that our BPA sensing device produces image pairs consisting of grayscale and laser images. Collecting those images is challenging because there is a

misalignment between those image pairs images. The misalignment occurs since there is only one image sensor that introduces a delay between the two frames. We need to remove or minimize this misalignment since we use both types of images in our final CNN model. To remove this misalignment, we have to fix the camera till image pair is captured. Therefore, we collect images by capturing each frame individually by fixing the camera because using the streaming video to capture images does not work.

In the purpose of taking many images and to speed up the capturing process, we modified the python SDK code that generates the frames from the camera. So the code always re-runs after capturing each paired image. Re-running the code takes some seconds with a notification sound, that gives us time and notification to change the position of the camera to capture new images. We fix the camera by putting it on a tripod while capturing. We put the tripod on a serving trolley, so it gives us many possibilities to move the camera when taking each frame. Figure 19 shows our capturing setup. Our procedure for taking each frame is as follows: 1) We start changing the position of the camera. 2) We fix the camera and waiting seconds till we hear the sound. 3) We change the position of the camera again and so on.

Figure 19. The image capturing setup.

In the capturing process, we take images for the selected materials from different angles and distances by moving the entire camera and tripod manually. Black curtains are used as background to avoid detecting the laser spots of materials that are not concerned. Most of the projected laser spots of this type of black curtains do not appear in the laser images if we fix the exposure parameters (Laser, Flood, and Gain) at specific values. We fixed the exposure parameters to the following values: (Laser =

4000 μs , Flood = 1500 μs, Gain = 18 dB) to avoid detecting the background spots and the saturation effect on the grayscale images.

5.2.3. Annotation Process

After finishing the data collection, the next stage is annotating the collected images. Annotation involves drawing a polygon approximation along the boundaries of the objects as the Figure 21 shows. Hence, a tool with a graphical user interface is needed to perform the annotation. The annotation tool that we utilise for the project is called Labelbox. The annotation result of this tool can be exported as a JSON file.

We create a python script to generate masks from polygons, which are saved to the JSON file. The generated masks are so important because they will represent as ground truth (GT) for our CNN model. Our annotation process depends on the grayscale images. Because those images look challenging to identify the objects, we used the contrast stretching technique to get clear edges and boundaries of the objects to facilitate the annotation process. Contrast stretching is a simple image enhancement technique that tries to improve the contrast in an image by stretching the range of intensity values to extend them to a desired range of values. The total time of the annotation process was around 122 hours. The average time for each image was 3.2 minutes. Other statistics about our annotation process is shown in Figure 20 an Table 5.

Table 5. the count and share of the labelled objects for each class.

Object	Count	Share (approx)
Banana	1849	21%
Apple	1767	20%
Paper	1662	19%
Orange	1336	15%
Potato	847	9%
Cotton	499	6%
Wood	428	5%
Broadcard	198	2%
Aluminum	149	2%
Skin	50	1%

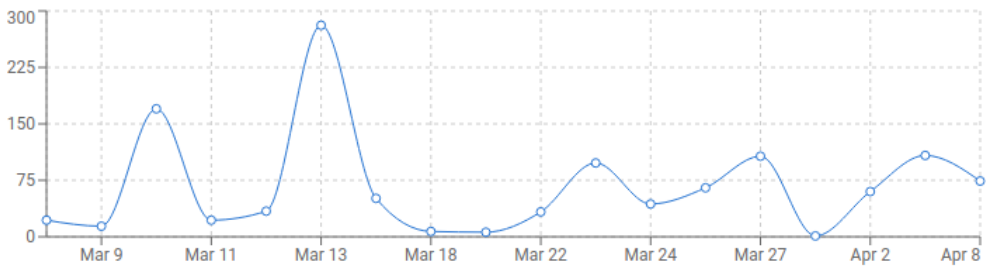

Figure 20. The count of the labelled objects versus time (days).

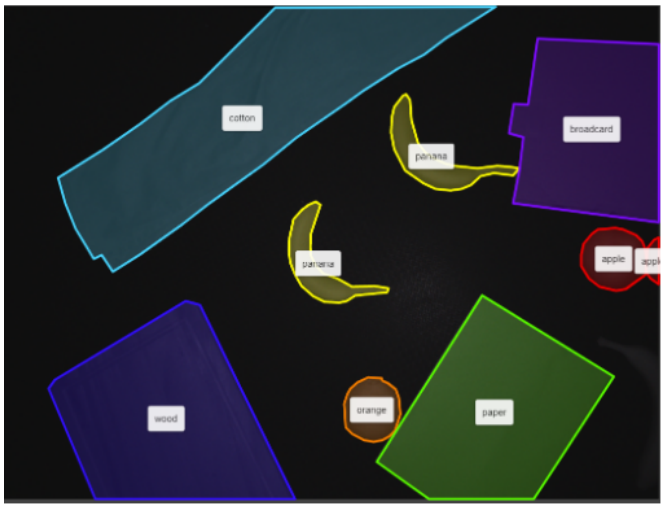

Figure 21. The count of the labelled objects versus time (days).

5.3. Data Preparation and Validation

Our dataset focuses on the local appearance (local image context) of materials and tries as possible to reduce the global context information like what object it makes up or where the material is, can be crucial to recognize materials. The materials included in this dataset have different shapes and forms like orange, apple, and potato have approximately the same spherical shape. The same thing happens for board-card, wood, and paper have a rectangular shape. However, we don't deny of availability of other global context information that the CNNs can learn from it like colour and some specific shapes like the banana shape. In general, we did all our best to focus on materials characteristics and not objects.

The dataset has two types of images, which are the grayscale and laser images, in addition to the ground truth (GT). The dataset has about 2300 samples of images where each image in the dataset comes at a 1080 x 1440 resolution. The dataset was split into around 70% training, 25% validation and 5% test datasets as Table 6 shows. Figure 22 shows examples of the dataset.

Table 6. The classes of the dataset with its RGB and index value.

Training	Validation	Testing
1650	550	100

The grayscale images were saved as 3 channel RGB images in JPG format with a 24 bit unsigned integer depth. The laser images were copied from 1 channel to 3 channels, and saved in JPG format with a 24 bit unsigned integer depth. The GT images were saved as 3 channel RGB images in PNG format, where the background class of the GT images is black RGB(0,0,0) and the other classes have different colours as Table 7 shows. In order to decrease the number of dimensions our CNN model has to process, we convert each RGB colour in the GT images to an indexed colour value, for example, the apple class RGB(255,0,0) to (2). All the values are shown in Table 7. Finally, we saved the dataset as TensorFlow Records format (tfrecord or TFR) that makes storing training data more efficient.

Table 7. The classes of the dataset with their RGB and index values.

Class Number and Value	Class Name	RGB Value
1	Background	(0,0,0)
2	Apple	(255,0,0)
3	Orange	(255,165,0)
4	Banana	(255,255,0)
5	Paper	(127,255,0)
6	Aluminium	(135,206,235)
7	Potato	(0,250,154)
8	Cotton	(30,144,255)
9	Wood	(205,133,63)
10	BoardCard	(128,0,128)
11	Skin	(255,192,203)

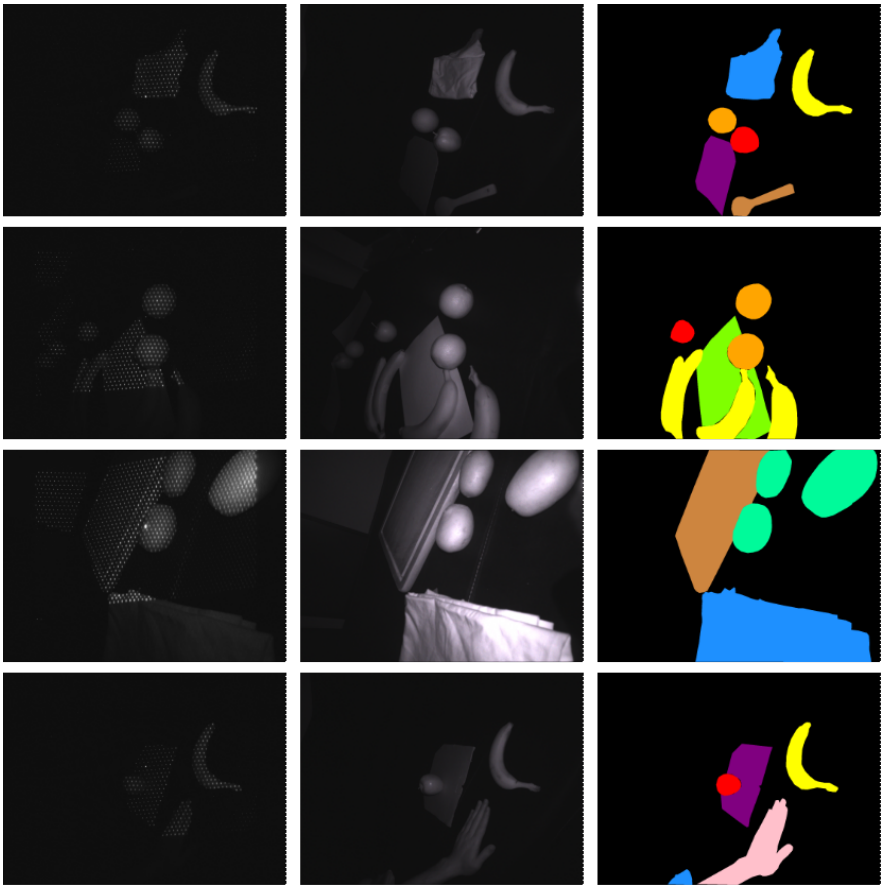

Figure 22. Some samples of the dataset: on the left side are laser images, on the middle side are the grayscale images, and on the right side are the GT.

5.4. Model Building

We adopt two different types of architectures in this work. Our first proposed architecture belongs to the family of the encoder-decoder approaches, as illustrated in Figure 23. The model is based on DeepLabv3+, where the encoder module gradually

degrades the feature maps and catches higher semantic information, and the decoder module gradually recovers the spatial information.

At the encoder level, the encoder applies atrous convolutions to extract the features computed by CNNs at an arbitrary resolution. We adopt output stride = 16 for denser feature extraction by removing the striding in the last block and applying the atrous convolution correspondingly. Furthermore, the encoder augments the ASPP module, which uses convolutional features at multiple scales with filters at multiple sampling rates. For the feature extraction part, the proposed model uses ResNet as its network backbone that allows us to train different deeper versions of the same architecture, which will provide a consistent metric gain over the addition of more layers.

At the Decoder level, we use a simple yet effective decoder module, where the encoder features are first bi-linearly upsampled by a factor of 4 and then concatenated with the corresponding low-level features from the network backbone that have the same spatial resolution. Before the concatenation, we apply 1 x 1 convolution on the low-level features to decrease the number of channels. After concatenation, we apply 3 x 3 convolutions to improve the features followed by simple upsampling by a factor of 4.

On the other hand, our second proposed architecture as illustrated in Figure 24 is the same as the first architecture except for some modification in the input types where the input grayscale image is concatenated with its corresponding laser image to form a 4 channel tensor rather than 3 channels the first proposed architecture.

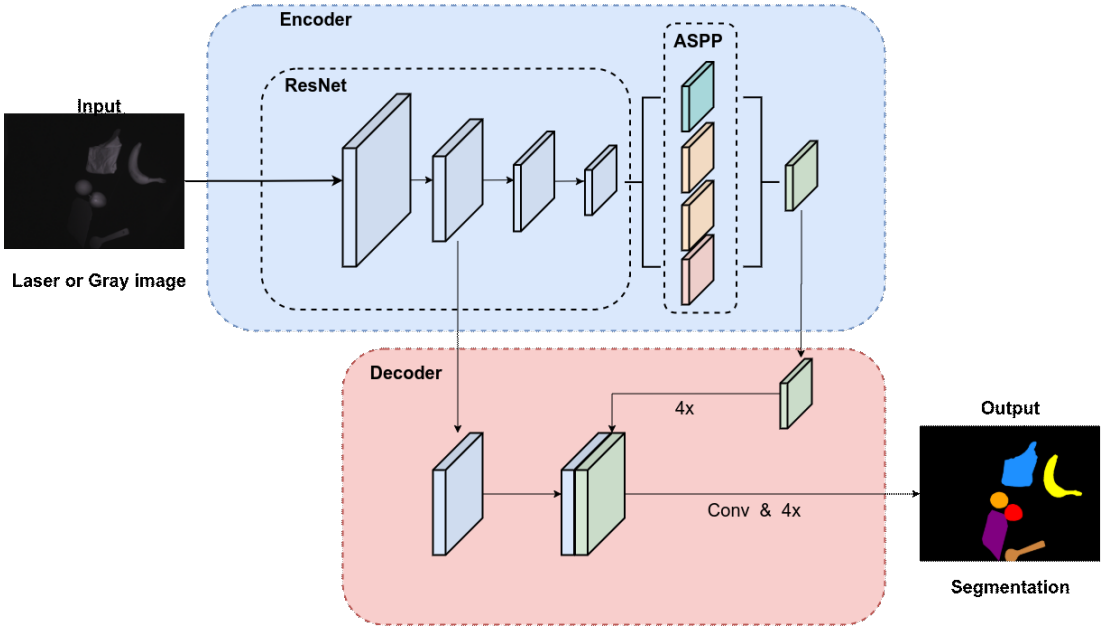

Figure 23. The first proposed architecture.

5.5. Training

We primarily concentrate on two research questions: (I) how to obtain segmented material maps; and (II) how the laser spots contribute and to the segmentation results?

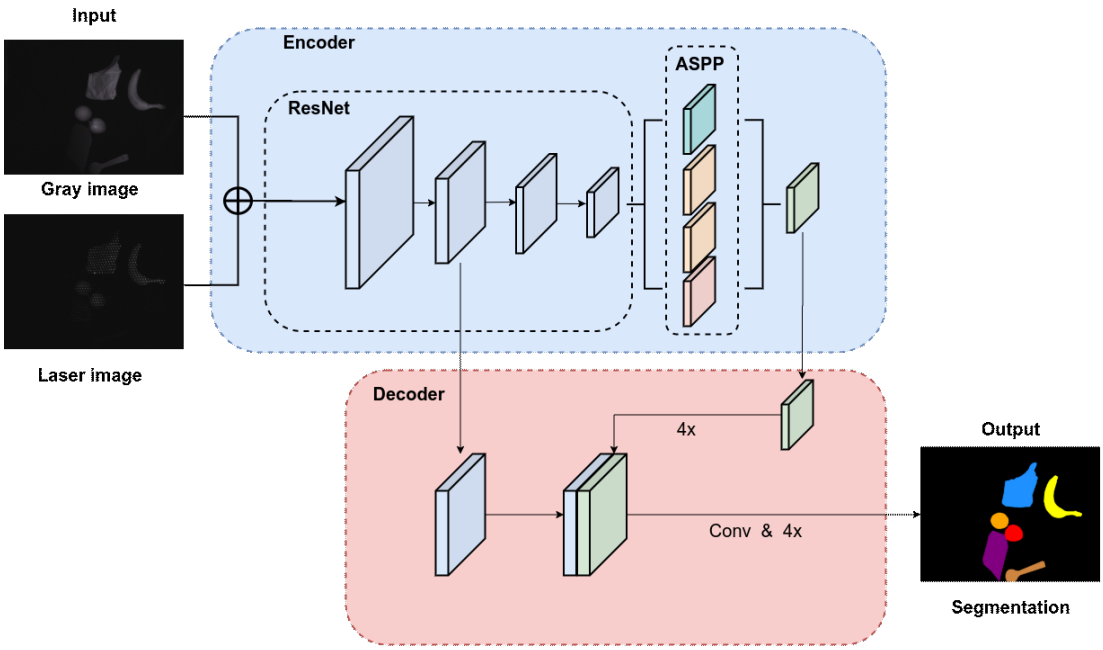

Figure 24. The second proposed architecture.

Therefore, the main focus of our work is to offer segmented material maps with a comparative analysis across different types of architectures to observe how the laser spots contribute to the whole segmentation. We built our experiments on 3 main types of models, and for each model, we apply different types of backbone architectures, as described in the following:

Gray Model: We use only grayscale images as input to the architecture shown in Figure 23. We implement ResNet-18, ResNet-34 as different backbones for the network to see the effect of the network depth on the results. Both ResNet-18 and ResNet-34 models have been trained from scratch.

Laser Model: We use only laser images as input to the architecture shown in Figure 23. We implement ResNet-18, ResNet-34, and ResNet-50 as different backbones for the network to see how the network depth would effect on the results. ResNet-18 and ResNet-34 models have been trained from scratch while ResNet-50 network was fine-tuned from the pre-trained ResNet-50 network, which has been trained on the ImageNet dataset.

GrayLaser Model: We use both grayscale and laser images as input to the architecture shown in Figure 24. We implement ResNet-18, ResNet-34, and ResNet-50 as different backbones for the network. ResNet-18 and ResNet-34 models have been trained from scratch. ResNet-50 network was fine-tuned from the pre-trained ResNet-50 network, which has been trained on the ImageNet dataset. This would go beyond ordinary transfer learning because we train the model to take in new input features. The solution is to expand the convolution filters in the first layer so that they have 4 channels instead of regular 3 channels (RGB channels). The RGB channels (first 3 channels) of these filters are initialized with the ResNet-50 pretrained weights, while the new channels start learning from scratch.

We perform all our experiments in TensorFlow [55], and train using Stochastic Gradient Descent (SGD) with a momentum of 0.9, weight decay of 0.0001 and

adaptive learning rates. For all networks, we start with the initial learning rate of 7e-3. We keep batch norm statistics frozen during the training. The loss function used is the sum of cross-entropy terms for each spatial position in the CNN output map.

For benchmarking, we use a workstation with 32 GB RAM with Intel Core i7 @2.6GHz processor, and 3 types of GPUs that are NVIDIA GeForce RTX 2070, NVIDIA T4, and NVIDIA P100 GPUs. We conduct our experiments on our generated dataset that was discussed before. This dataset comprises of 2300 samples of grayscale and laser images with 11 segmented class labels, of which 1650 are used for training and 550 for testing, respectively. We keep training until 150 epochs with different batch sizes depending on the network, and we use the poly learning rate policy [68], where the initial learning rate is multiplied by $(1 - \frac{iter}{max_iter})^{power}$ with power = 0.9 and the number of training iterations is 30000 for all experiments.

5.6. Results

5.6.1. Metrics

In our experimentation, Mean Intersection over Union (MIoU) and Pixel accuracy (PA) metrics were used to measure how accurate the result is compared to the Ground truth (GT) data. Mean Intersection over Union (MIoU) is the area of overlap between the predicted segmentation and the GT divided by the area of union between the predicted segmentation and the ground truth. MIoU is calculated using the following equation:

$$MIoU = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{TP + FP + FN} \tag{8}$$

where:n is the number of classes, TP is true positive pixels, FP is false positive pixels, FN is false negative pixels. Another metric, pixel accuracy (PA) is the percent of pixels in the image that are classified correctly. PA is calculated using the following equation:

$$PA = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

5.6.2. Evaluation Results

Our analysis is based on quantitative and quantitative results.

Quantitative results:

- At the first level, our quantitative results on validation sets that are based on Mean IoU, and the training time PA are given in Table 8 for all models (Gray, Laser, and GrayLaser) with different networks that were implemented in this work.
- At the second level, we compare the mIoU for each class on ResNet-18 and ResNet-34 network for all models: Gray, Laser, and GrayLaser. The results are shown in Table 9.

- At the third level, we compare the mIoU for each class on ResNet-50 network for Laser and GrayLaser. The results are shown in Table 10.

Qualitative results:

- At the first level, we visualize some results of both networks of Gray model (ResNet-18 and ResNet-34) with Ground truth (GT) images. The results are shown in Figure 25.
- At the second level, we visualize some results of all networks of Laser model (ResNet-18, ResNet-34, and ResNet-50) with Ground truth (GT) images. The results are shown in Figure 26.
- At the third level, we visualize some results of all networks of GrayLaser model (ResNet-18, ResNet-34, and ResNet-50) with Ground truth (GT) images. The results are shown in Figure 27.
- At the fourth level, we visualize results of some testing samples for ResNet-18 based model of Gray, Laser, and GrayLaser architectures. The results are shown in Figure 28.
- At the fifth level, we visualize results of some testing samples for ResNet-34 based model model of Gray, Laser, and GrayLaser architectures. The results are shown in Figure 29.
- At the sixth level, we visualize results of some testing samples for ResNet-50 based model model of Gray, Laser, and GrayLaser architectures. The results are shown in Figure 30.

Table 8. Performance comparison between all implemented models.

Type of model	Architecture	PA	MIoU	Training Time
Gray	ResNet-18	95.56%	40.18%	9h
Gray	ResNet-34	93.66%	31.30%	11h
Laser	ResNet-18	95.75%	52.14%	9h
Laser	ResNet-34	94.77%	41.87%	10h
Laser	ResNet-50	98.93%	85.38%	15h
GrayLaser	ResNet-18	97.09%	62.91%	14h
GrayLaser	ResNet-34	96.43%	51.52%	18h
GrayLaser	ResNet-50	99.52%	94.03%	20h

Table 9. Intersection over Union for each class on ResNet-18 and ResNet-34 of all models.

		ResNet-18			ResNet-34		
Class Number	Class Name	Gray IoU	Laser IoU	GL IoU	Gray IoU	Laser IoU	GL IoU
1	Background	97.69	97.32	98.85	95.10	96.79	98.70
2	Apple	29.18	40.26	48.72	36.72	34.76	42.13
3	Orange	46.39	46.63	55.63	46.34	42.63	50.93
4	Banana	45.63	52.45	65.75	45.46	48.61	61.46
5	Paper	67.67	63.96	72.49	51.91	55.04	70.17
6	Aluminium	03.84	28.61	61.65	04.15	18.41	31.51
7	Potato	35.74	42.28	56.96	26.61	29.13	39.19
8	Cotton	47.59	51.09	58.23	16.19	36.64	46.77
9	Wood	46.34	46.74	53.26	14.05	28.45	41.90
10	BoardCard	16.17	68.86	71.94	07.05	50.95	55.26
11	Skin	05.79	35.37	48.55	00.74	19.15	28.71

Table 10. Intersection over Union for each class on ResNet-50 of Laser and GrayLaser models.

Class Number	Class Name	Laser IoU	GrayLaser IoU
1	Background	98.98	99.56
2	Apple	85.23	90.39
3	Orange	85.09	90.92
4	Banana	85.81	92.12
5	Paper	92.78	96.31
6	Aluminium	62.94	92.80
7	Potato	90.23	96.55
8	Cotton	92.41	97.76
9	Wood	92.86	96.81
10	BoardCard	85.08	96.90
11	Skin	67.81	84.16

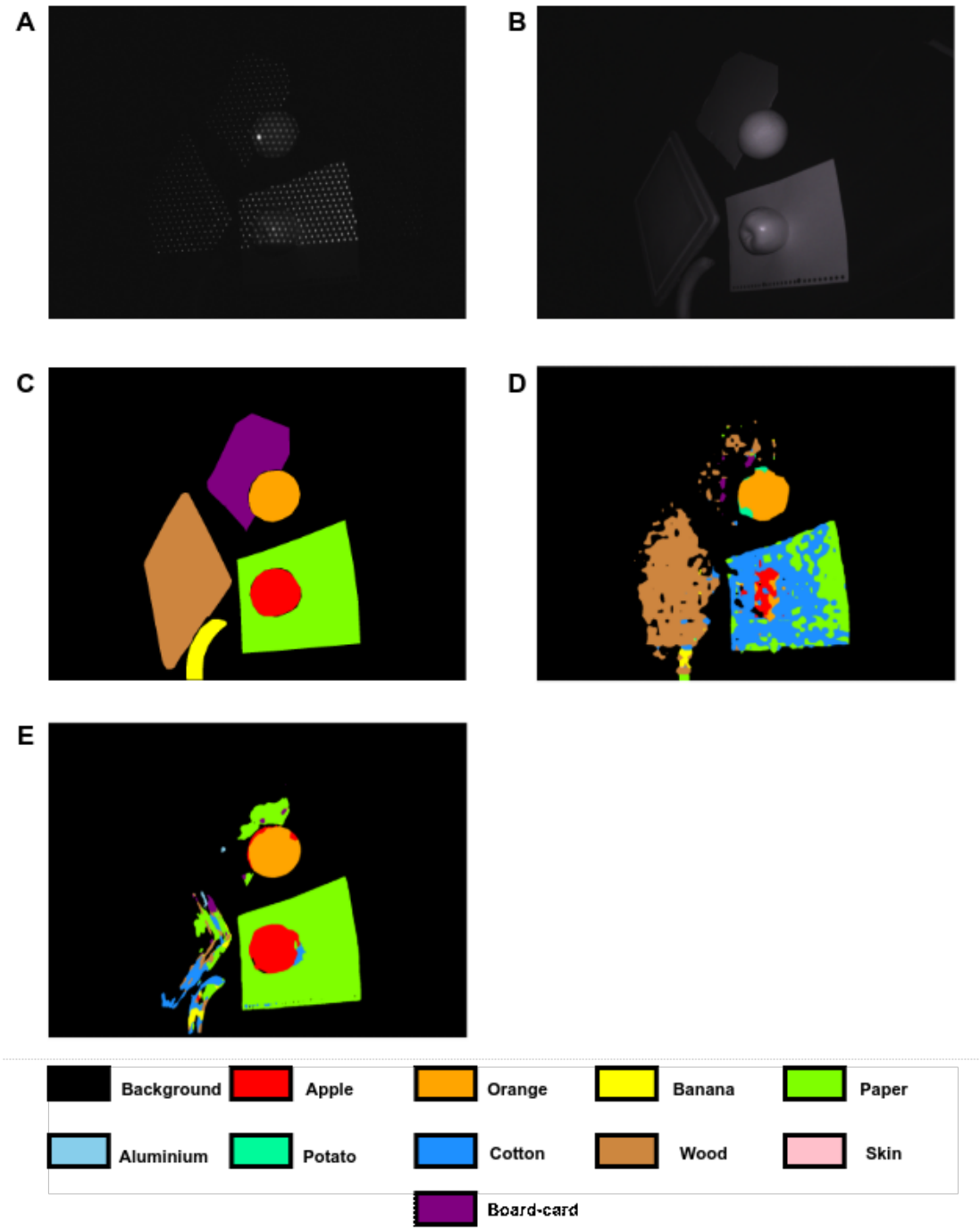

Figure 25. Visualisation results of Gray model from one testing sample. A:laser image. B:Grayscale image. C:Ground Truth GT from our generated dataset. D:Result of ResNet-18 based model. E:Result of ResNet-34 based model.

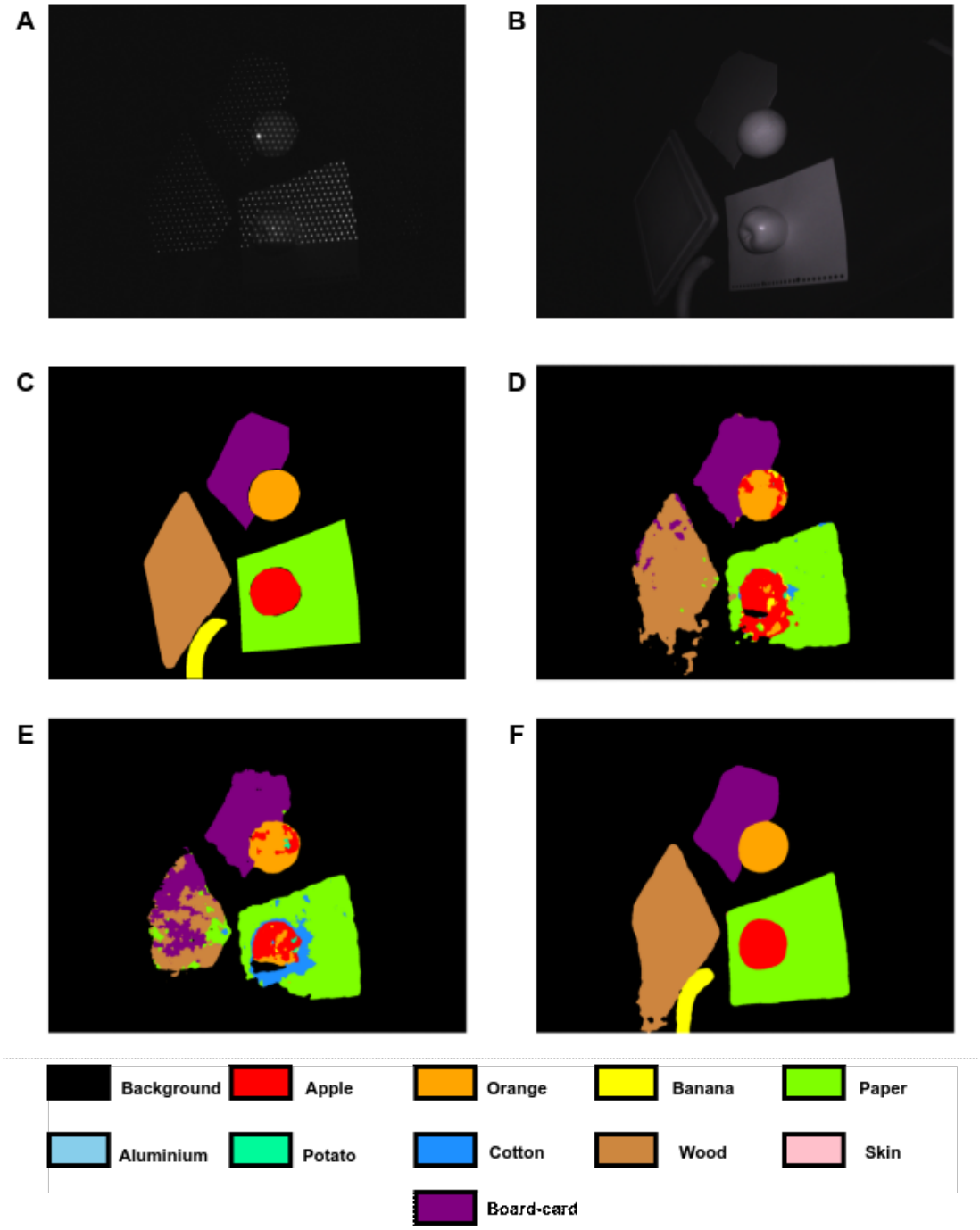

Figure 26. Visualisation results of Laser model from one testing sample. A:Laser image. B:Grayscale image. C:Ground Truth GT from our generated dataset. D:Result of ResNet-18 based model. E:Result of ResNet-34 based model. F:Result of ResNet-50 based model with transfer learning.

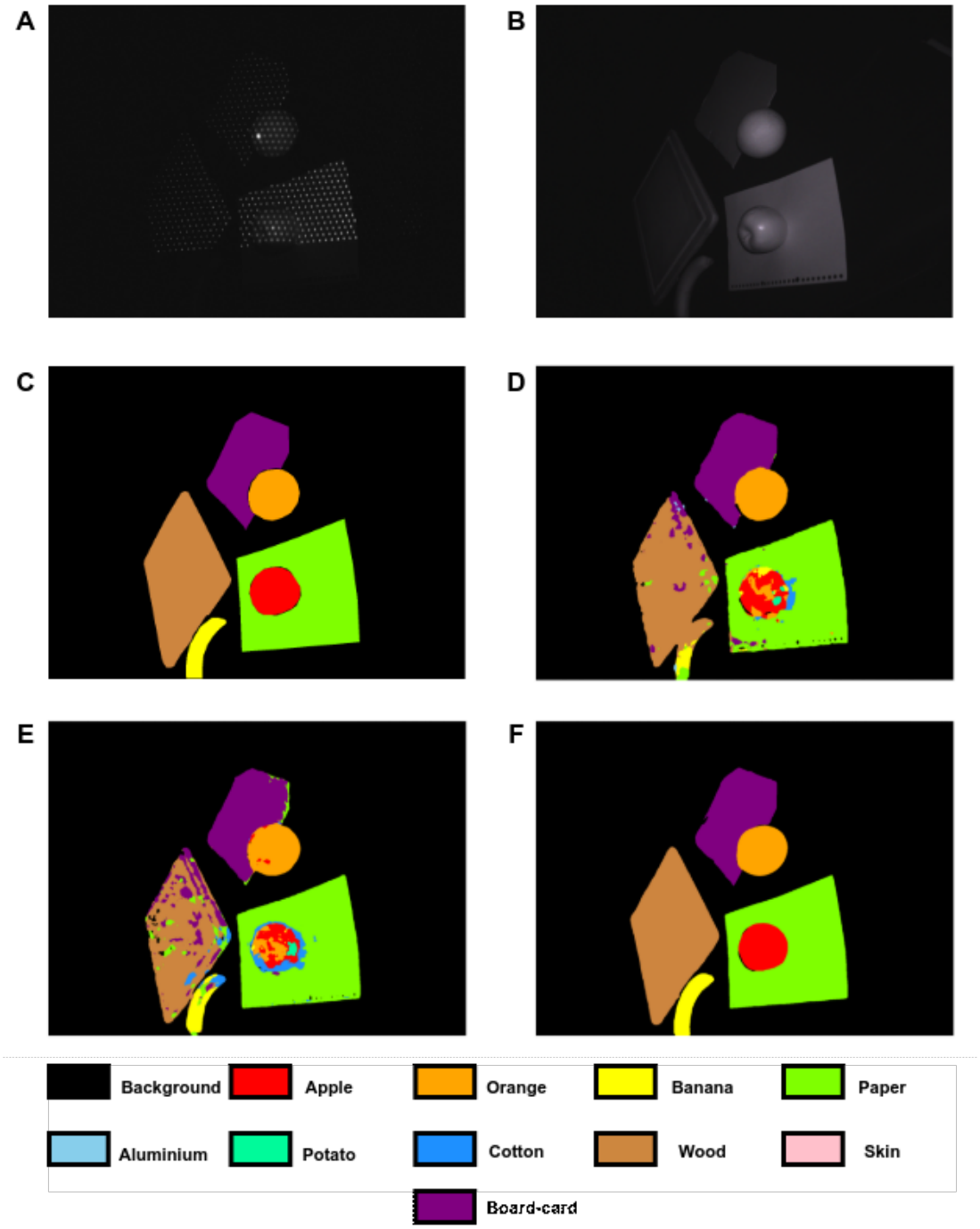

Figure 27. Visualisation results of GrayLaser model from one testing sample. A:Laser image. B:Grayscale image. C:Ground Truth GT from our generated dataset. D:Result of ResNet-18 based model. E:Result of ResNet-34 based model. F:Result of ResNet-50 based model with transfer learning.

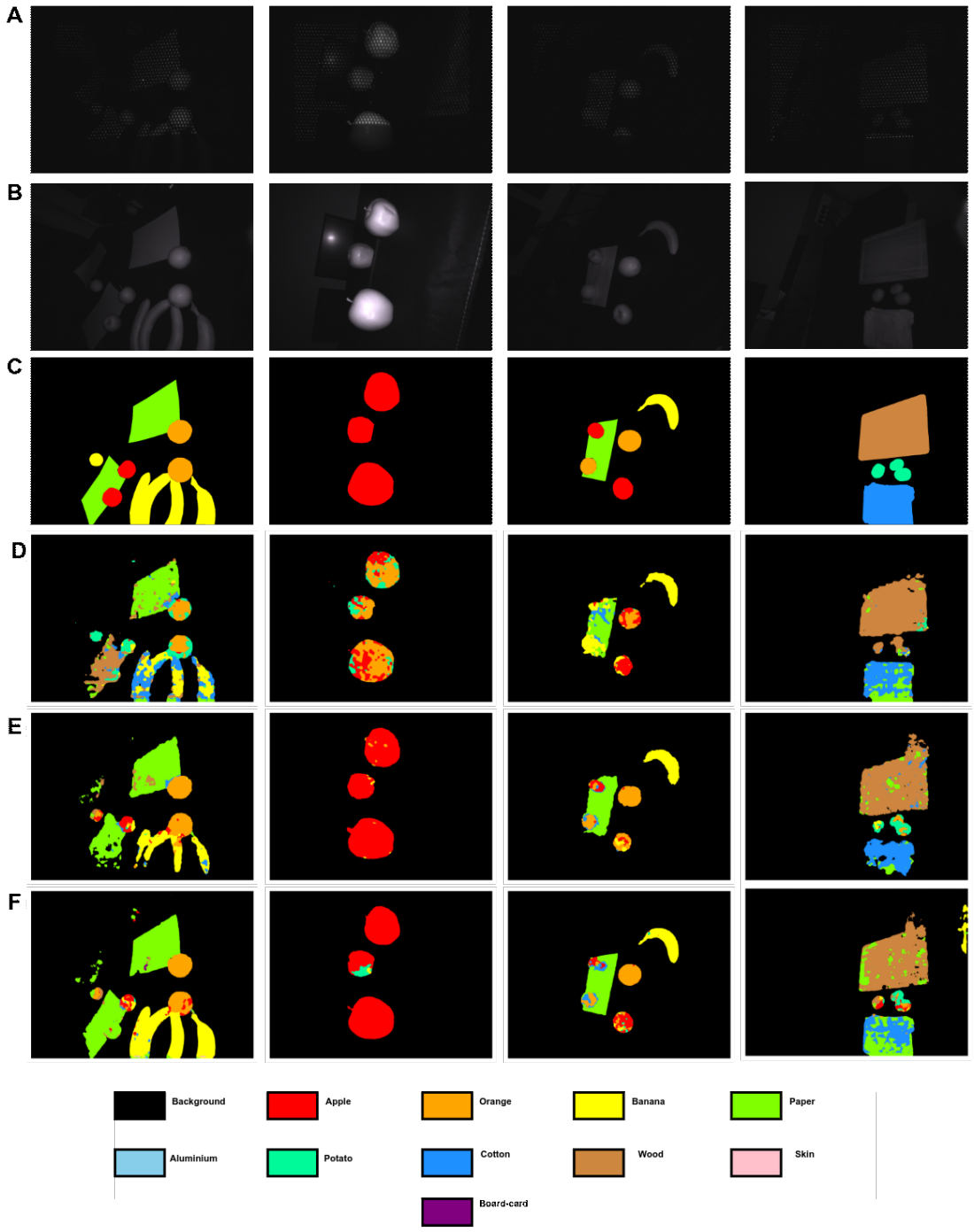

Figure 28. Visualisation results of ResNet-18 based models from one some testing samples. A:Laser image. B:Grayscale image. C:Ground Truth GT from our generated dataset. D:Result of Gray model. E:Result of Laser model. F:Result of GrayLaser model.

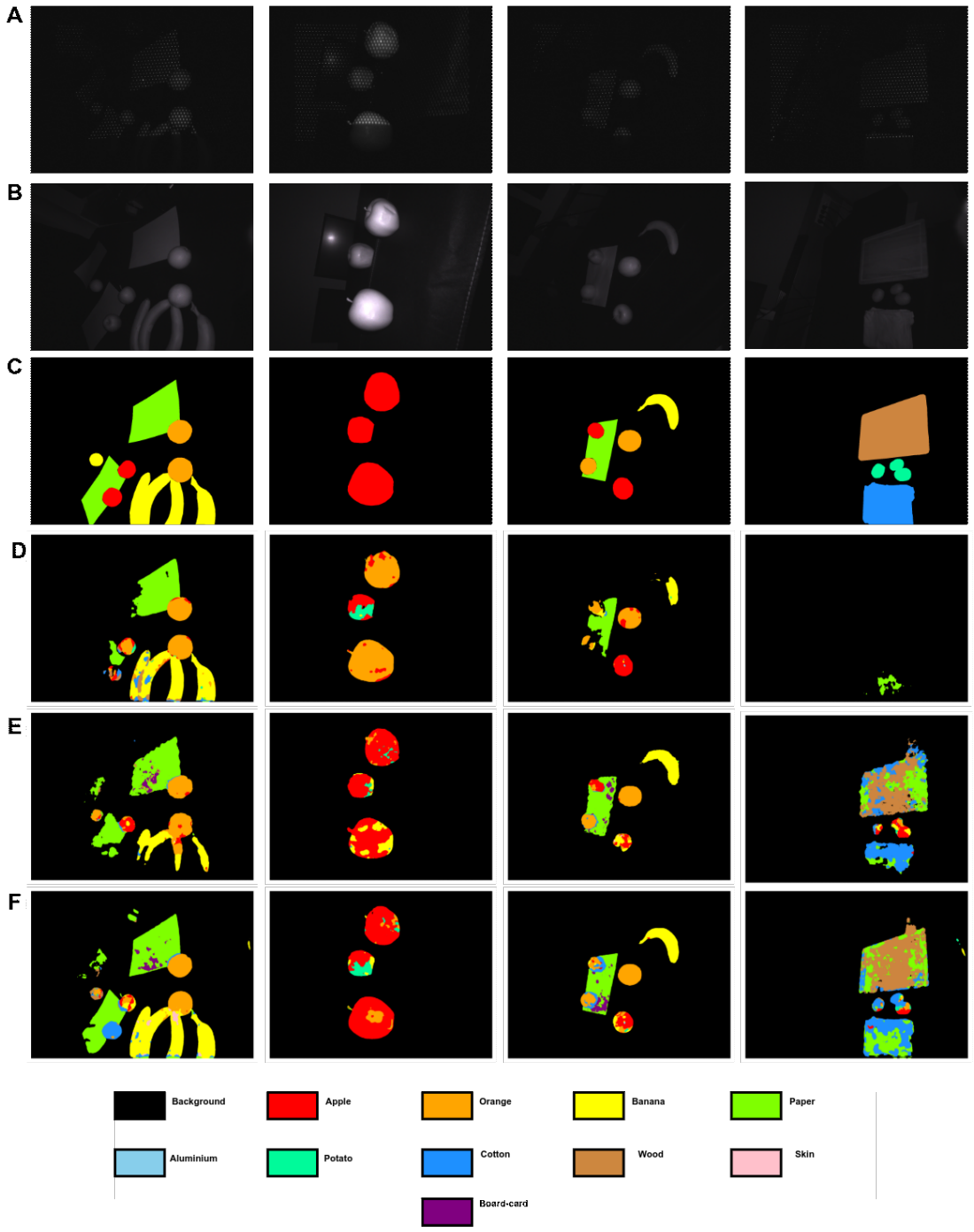

Figure 29. Visualisation results of ResNet-34 based models from one some testing samples. A:Laser image. B:Grayscale image. C:Ground Truth GT from our generated dataset. D:Result of Gray model. E:Result of Laser model. F:Result of GrayLaser model.

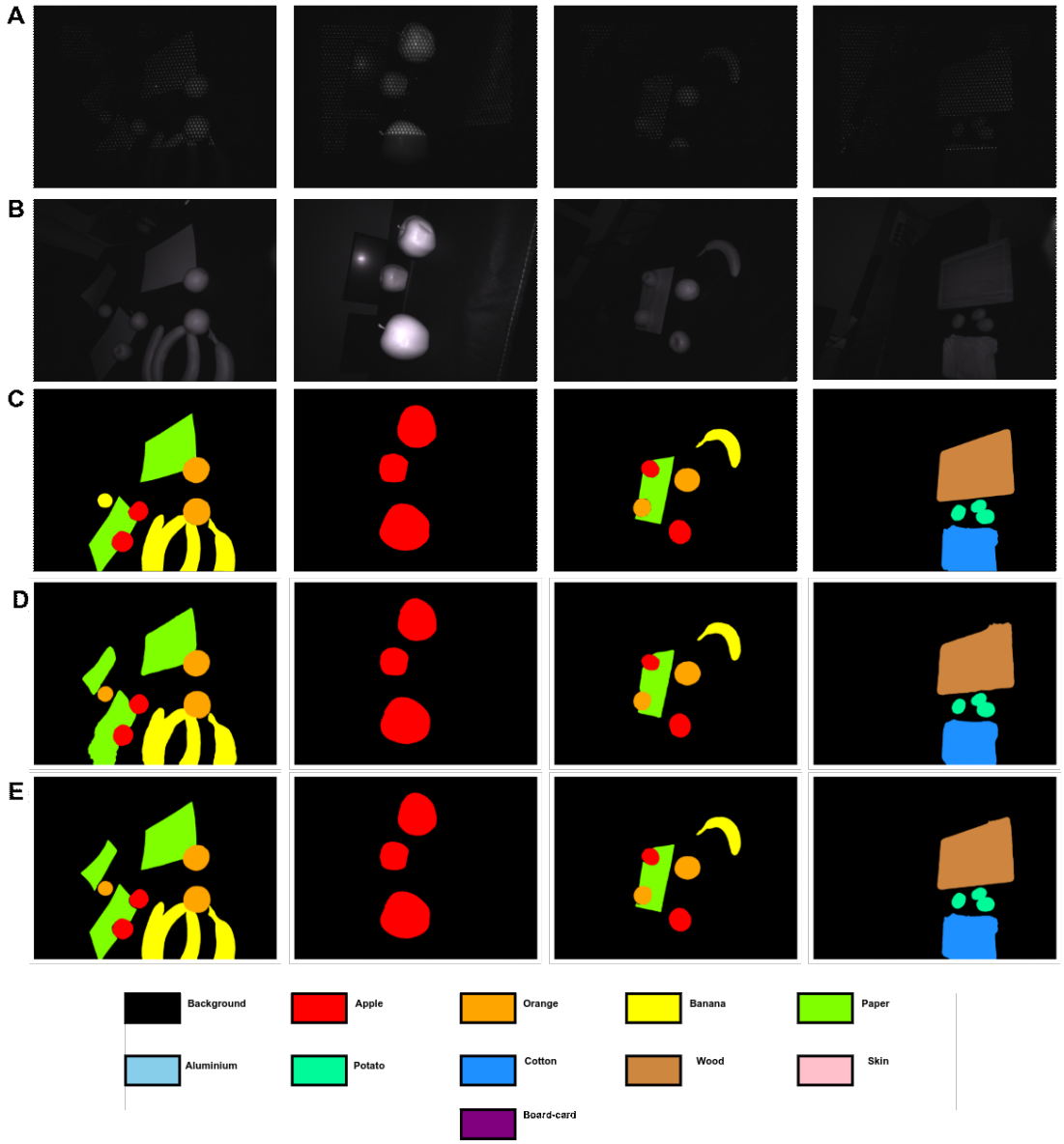

Figure 30. Visualisation results of ResNet-50 based models with transfer learning from one some testing samples. A:Laser image. B:Grayscale image. C:Ground Truth GT from our generated dataset. D:Result of Laser model. E:Result of GrayLaser model.

6. DISCUSSION

There are quite a few interesting observations that can be made from the results. So we will discuss our quantitative and qualitative results on how the laser spots affect the performance of a convolutional neural network.

6.1. Depends on the Quantitative Results

We first start by analysing Table 8, which summarise all the results of the implemented architectures in this work. It is very clear that the networks of the GrayLaser model type gives the best results compared to Laser model and Gray model with the same backbone network. So from ResNet-18 networks, we see that GrayLaser model gives the best results by achieving an MIoU of 62.91% and PA 97.09% and in the second place is the Laser model achieving an MIoU of 52.14% and 95.75% PA, and the Gray model with 40.18% MIoU and 95.56% PA. The second observation from this cooperation is that the difference of MIoU is about 10% between GrayLaser model and Laser, and about 12% between Laser and Gray model. We see the same arrangement if we compare the results, on ResNet-34 networks. The GrayLaser model is first, then Laser model, and finally Gray, with a difference of MIoU is about 10% between GrayLaser model and Laser, and also 10% between Laser and Gray model.

Our explanation of the decreasing performance of all the models when we used ResNet-34 which is deeper network than ResNet-18 is the lack of data, and it is not that deeper CNNs perform worse because Deep CNNs have more degrees of freedom (DoF) and each layer can be seen as a cognitive step. So we need a large dataset to properly train deeper CNNs. However, the arrangement of which model do the best performance still does not change.

The ResNet-50 network of the GrayLaser model has the top performance with MIoU of 94.03% and 99.52% PA. This result is not surprising since the weights are pretrained on the huge ImageNet dataset and it is a deeper network than ResNet-18 and Resnet-34, which were trained from scratch. While the network of the Laser model that has the same deep network (ResNet-50) achieved 85.38% MIoU and 98.93% PA which is less than the GrayLaser model by around 9% MIoU and 0.5% PA. Here we want to mention that we always get high PA accuracy because the background class dominate the image and it is well predicted by our models so the PA increase while other classes cover some small portion of the image. Thus the small variation in PA really matters but we mainly focus on MIoU metric, which helps us clearly see the difference between the performance.

From Table 9 and 10 that show the IoU accuracy for each class, we can observe again that the GrayLaser model has the best accuracy for each class among the 11 classes compared to the Laser and Gray. As a second observation from Table 9 and 10, we see that the "skin" and "aluminium" classes have lower accuracy at Gray and Laser models where Gray model is the worse and not even close to other two models. The reason behind this result is perhaps because of low availability of "skin" and "aluminium" data on the dataset compared to the other classes, but those classes have similar accuracy to other classes or even better than some of them on the GrayLaser model. However,

the "apple" class achieved low accuracy compared to other classes although its data is abundance in the dataset.

6.2. Depends on the Qualitative Results

From Figure 25, 28 and 29 that represents the visualisation results of the Gray networks, it is clear that gray images do proper recognition of the boundaries and edges of materials included in the image. However, it is not good at all in our main purpose of segmenting and recognising materials.

From Figure 26, 28, 29 and 30 that represents the visualisation results of the Laser networks, we can see that the laser images are doing well in recognising materials due to the laser spot characteristics. However, we can see from images that some parts of materials were cut off because of the limited area that the laser spots cover. Those cut parts have an effect on the final results because of missing laser spot information.

From Figure 27, 28, 29 and 30 that represents the visualisation results of the GrayLaser networks, we see that using laser and gray images together gives better results in recognising materials and produces good segmentation and identifying the edges of the materials inside the images. Therefore we can see that the ability of recognising and segmenting materials came mainly from the laser images (which means laser spots). At the same time, this model gets the information of the missing areas that the laser images have from the gray images.

We conclude our discussion of this thesis in four main points that are as follows:

- All those results and observation support the idea that the laser spots have significant contribution on material and object segmentation results.
- Using laser and gray images provided from our BPA sensing device is the best solution because it combines the advantages and features of the laser and gray images.
- Using our moderate-sized dataset that focuses on the local appearance of materials and tries to eliminate the global context information, and with using a CNNs that were fed by both laser and gray images (or even only laser images) produces good segmentation results.
- Using a pretrained model really helps for initialising the network weights that produces excellent results at the final stage.

6.3. Limitations

Although our generated dataset was enough to train the implemented networks, it is considered medium or small compared to the current existing dataset in computer vision. As a second point about the dataset, we have to take into consideration the error during forming this dataset and its effect on the final results. Some of those errors and limitation include the incorrectly annotated pixels during the manual polygon approximation process, the limitation of the existing misalignment between laser and

gray images that was discussed previously in this thesis. Finally, we also have to mention the requirement of computational resources because some of the top methods in computer vision require heavy usage of near-supercomputers for the training phase which is not available in our context.

6.4. Future Work and Improvements

For improving the segmentation results, we suggest refining the collection of images by capturing only objects with a black background and the objects have to take most of the image size (no small objects), and in addition removing images that have far view and noisy background from the training data. We also suggest testing the option that we feed the handcrafted features that the BPA generate, as a tensor to the network along with the image information for guiding the segmentation. That would give some kind of a baseline. After that, we could take the patches covered by the laser spots and use them as inputs directly, but in that case, we probably need more training data. Finally as a good way of improving the performance one should try to find solutions to the limitations mentioned above.

7. CONCLUSION

We introduce a new pixel-wise material segmentation dataset in this thesis. Our dataset is the first one collected with BPA sensing camera for semantic segmentation task, and it contains 2300 high-resolution images, where the grayscale images have been annotated and labelled to the material of interest. We exploited the recent success in DL approaches, and trained many CNNs on this dataset to perform pixel-wise material segmentation. Additionally, we performed a detailed investigation into the ideal input type provided by the BPA sensing solution in material recognition. The experimental results conclusively demonstrate that pixel-wise material segmentation based on only the explicit integration of the local appearance of the laser spots achieves improved accuracy. These results constitute important baselines that can encourage further research in the use of CNNs for other material recognition and segmentation applications. Our dataset also enables other similar or novel applications of pixel-wise material segmentation.

8. REFERENCES

[1] Sharma A.K. & Kumari K. (2017) Human depth perception. International Journal of Advance Research, Ideas and Innovations in Technology 3, pp. 864–869.

[2] Kim S., Nam J. & Ko B. (2019) Fast depth estimation in a single image using lightweight efficient neural network. Sensors 19, p. 4434.

[3] Sanz P.R., Mezcua B.R. & Pena J.M.S. (2012) Depth estimation-an introduction. In: Current Advancements in Stereo Vision, IntechOpen.

[4] He Y. & Chen S. (2018) Advances in sensing and processing methods for three-dimensional robot vision. International Journal of Advanced Robotic Systems 15, p. 1729881418760623.

[5] Gandhi V., Čech J. & Horaud R. (2012) High-resolution depth maps based on tof-stereo fusion. In: 2012 IEEE International Conference on Robotics and Automation, IEEE, pp. 4742–4749.

[6] Chen C.S., Hung Y.P., Chiang C.C. & Wu J.L. (1997) Range data acquisition using color structured lighting and stereo vision. Image and Vision Computing 15, pp. 445–456.

[7] Wei B., Gao J., Li K., Fan Y., Gao X. & Gao B. (2009) Indoor mobile robot obstacle detection based on linear structured light vision system. In: 2008 IEEE International Conference on Robotics and Biomimetics, IEEE, pp. 834–839.

[8] Johnson-Roberson M., Bryson M., Friedman A., Pizarro O., Troni G., Ozog P. & Henderson J.C. (2017) High-resolution underwater robotic vision-based mapping and three-dimensional reconstruction for archaeology. Journal of Field Robotics 34, pp. 625–643.

[9] Silberman N. & Fergus R. (2011) Indoor scene segmentation using a structured light sensor. In: 2011 IEEE international conference on computer vision workshops (ICCV workshops), IEEE, pp. 601–608.

[10] Song T.H. & Ha J.E. (2016) Visual surveillance using wide-angle camera and laser range finder. Electronics Letters 52, pp. 445–447.

[11] Zhang Z. (2012) Microsoft kinect sensor and its effect. IEEE MultiMedia 19, pp. 4–10.

[12] Han J., Shao L., Xu D. & Shotton J. (2013) Enhanced computer vision with microsoft kinect sensor: A review. IEEE Transactions on Cybernetics 43, pp. 1318–1334.

[13] Marzani F., Voisin Y., Voon L.F.L.Y. & Diou A. (2002) Calibration of a three-dimensional reconstruction system using a structured light source. Optical Engineering 41, pp. 484–493.

[14] Dipanda A. & Woo S. (2005) Towards a real-time 3d shape reconstruction using a structured light system. Pattern recognition 38, pp. 1632–1650.

[15] Bhoi A. (2019) Monocular depth estimation: A survey. arXiv preprint arXiv:1901.09402 .

[16] Fu H., Gong M., Wang C. & Tao D. (2017) A compromise principle in deep monocular depth estimation. arXiv preprint arXiv:1708.08267 .

[17] Smolyanskiy N., Kamenev A. & Birchfield S. (2018) On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1007–1015.

[18] Zhao C., Sun Q., Zhang C., Tang Y. & Qian F. (2020) Monocular depth estimation based on deep learning: An overview. arXiv preprint arXiv:2003.06620 .

[19] Zhang S. (2013) Handbook of 3D machine vision: Optical metrology and imaging. CRC press.

[20] Chen Z., Sun X., Wang L., Yu Y. & Huang C. (2015) A deep visual correspondence embedding model for stereo matching costs. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 972–980.

[21] Flynn J., Snavely K., Neulander I. & Philbin J. (2018), Deepstereo: learning to predict new views from real world imagery. US Patent 9,916,679.

[22] Seki A. & Pollefeys M. (2017) Sgm-nets: Semi-global matching with neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 231–240.

[23] Schonberger J.L., Sinha S.N. & Pollefeys M. (2018) Learning to fuse proposals from multiple scanline optimizations in semi-global matching. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 739–755.

[24] Kendall A., Martirosyan H., Dasgupta S., Henry P., Kennedy R., Bachrach A. & Bry A. (2017) End-to-end learning of geometry and context for deep stereo regression. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 66–75.

[25] Chang J.R. & Chen Y.S. (2018) Pyramid stereo matching network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5410–5418.

[26] Deng L. & Yu D. (2014) Deep learning: methods and applications. Foundations and trends in signal processing 7, pp. 197–387.

[27] LeCun Y., Bengio Y. & Hinton G. (2015) Deep learning. nature 521, pp. 436–444.

[28] Simonyan K. & Zisserman A. (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

[29] Springenberg J.T., Dosovitskiy A., Brox T. & Riedmiller M. (2014) Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 .

[30] Graham B. (2014) Fractional max-pooling. arXiv preprint arXiv:1412.6071 .

[31] Zeiler M.D., Taylor G.W. & Fergus R. (2011) Adaptive deconvolutional networks for mid and high level feature learning. In: 2011 International Conference on Computer Vision, IEEE, pp. 2018–2025.

[32] Zeiler M.D. & Fergus R. (2013) Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557 .

[33] Hinton G.E., Srivastava N., Krizhevsky A., Sutskever I. & Salakhutdinov R.R. (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 .

[34] Glorot X. & Bengio Y. (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, pp. 249–256.

[35] Nair V. & Hinton G.E. (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814.

[36] Xu B., Wang N., Chen T. & Li M. (2015) Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853 .

[37] He K., Zhang X., Ren S. & Sun J. (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034.

[38] Srivastava N., Hinton G., Krizhevsky A., Sutskever I. & Salakhutdinov R. (2014) Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15, pp. 1929–1958.

[39] Ioffe S. & Szegedy C. (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 .

[40] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. & Rabinovich A. (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

[41] Chollet F. (2017) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251–1258.

[42] He K., Zhang X., Ren S. & Sun J. (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[43] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A.C. & Fei-Fei L. (2015) ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, pp. 211–252.

[44] Guo D., Fridriksson J., Fillmore P., Rorden C., Yu H., Zheng K. & Wang S. (2015) Automated lesion detection on mri scans using combined unsupervised and supervised methods. BMC medical imaging 15, p. 50.

[45] Seghier M.L., Ramlackhansingh A., Crinion J., Leff A.P. & Price C.J. (2008) Lesion identification using unified segmentation-normalisation models and fuzzy clustering. Neuroimage 41, pp. 1253–1266.

[46] Ess A., Müller T., Grabner H. & Van Gool L.J. (2009) Segmentation-based urban traffic scene understanding. In: BMVC, vol. 1, Citeseer, vol. 1, p. 2.

[47] Cordts M., Omran M., Ramos S., Rehfeld T., Enzweiler M., Benenson R., Franke U., Roth S. & Schiele B. (2016) The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3213–3223.

[48] Oquab M., Bottou L., Laptev I. & Sivic J. (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1717–1724.

[49] Everingham M. & Winn J. (2011) The pascal visual object classes challenge 2012 (voc2012) development kit. Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep .

[50] Brostow G.J., Fauqueur J. & Cipolla R. (2009) Semantic object classes in video: A high-definition ground truth database. Pattern Recognition Letters 30, pp. 88–97.

[51] Caesar H., Uijlings J. & Ferrari V. (2018) Coco-stuff: Thing and stuff classes in context. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1209–1218.

[52] Chen L.C., Papandreou G., Schroff F. & Adam H. (2017) Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 .

[53] Chen L.C., Zhu Y., Papandreou G., Schroff F. & Adam H. (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp. 801–818.

[54] Long J., Shelhamer E. & Darrell T. (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440.

[55] Bell S., Upchurch P., Snavely N. & Bala K. (2015) Material recognition in the wild with the materials in context database. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3479–3487.

[56] Huang C., Li Y., Change Loy C. & Tang X. (2016) Learning deep representation for imbalanced classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5375–5384.

[57] Guo D., Zhu L., Lu Y., Yu H. & Wang S. (2018) Small object sensitive segmentation of urban street scene with spatial adjacency between object classes. IEEE Transactions on Image Processing 28, pp. 2643–2653.

[58] Maire M., Stella X.Y. & Perona P. (2011) Object detection and segmentation from joint embedding of parts and pixels. In: 2011 International Conference on Computer Vision, IEEE, pp. 2142–2149.

[59] Plath N., Toussaint M. & Nakajima S. (2009) Multi-class image segmentation using conditional random fields and global classification. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 817–824.

[60] Boix X., Gonfaus J.M., Van de Weijer J., Bagdanov A.D., Serrat J. & Gonzàlez J. (2012) Harmony potentials. International journal of computer vision 96, pp. 83–102.

[61] Sermanet P., Eigen D., Zhang X., Mathieu M., Fergus R. & LeCun Y. (2013) Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 .

[62] Pinheiro P.H. & Collobert R. (2014) Recurrent convolutional neural networks for scene labeling. In: 31st International Conference on Machine Learning (ICML), CONF.

[63] Liu M.Y., Lin S., Ramalingam S. & Tuzel O. (2015) Layered interpretation of street view images. arXiv preprint arXiv:1506.04723 .

[64] Ronneberger O., Fischer P. & Brox T. (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer, pp. 234–241.

[65] Badrinarayanan V., Kendall A. & Cipolla R. (2017) Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence 39, pp. 2481–2495.

[66] Mohan R. (2014) Deep deconvolutional networks for scene parsing. arXiv preprint arXiv:1411.4101 .

[67] Brust C.A., Sickert S., Simon M., Rodner E. & Denzler J. (2015) Convolutional patch networks with spatial prior for road detection and urban scene understanding. arXiv preprint arXiv:1502.06344 .

[68] 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 and machine intelligence 40, pp. 834–848.

[69] Liu W., Rabinovich A. & Berg A.C. (2015) Parsenet: Looking wider to see better. arXiv preprint arXiv:1506.04579 .

[70] Chen L.C., Papandreou G., Kokkinos I., Murphy K. & Yuille A.L. (2014) Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 .

[71] Peng C., Zhang X., Yu G., Luo G. & Sun J. (2017) Large kernel matters–improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4353–4361.

[72] Adelson E.H. (2001) On seeing stuff: the perception of materials by humans and machines. In: Human vision and electronic imaging VI, vol. 4299, International Society for Optics and Photonics, vol. 4299, pp. 1–12.

[73] Sharan L., Rosenholtz R. & Adelson E. (2009) Material perception: What can you see in a brief glance? Journal of Vision 9, pp. 784–784.

[74] Zhang Y., Ozay M., Liu X. & Okatani T. (2016) Integrating deep features for material recognition. In: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, pp. 3697–3702.

[75] Wang T.C., Zhu J.Y., Hiroaki E., Chandraker M., Efros A.A. & Ramamoorthi R. (2016) A 4d light-field dataset and cnn architectures for material recognition. In: European Conference on Computer Vision, Springer, pp. 121–138.

[76] Iizuka S., Simo-Serra E. & Ishikawa H. (2016) Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Transactions on Graphics (ToG) 35, pp. 1–11.

[77] Everingham M., Eslami S.A., Van Gool L., Williams C.K., Winn J. & Zisserman A. (2015) The pascal visual object classes challenge: A retrospective. International journal of computer vision 111, pp. 98–136.

[78] Russell B.C., Torralba A., Murphy K.P. & Freeman W.T. (2008) Labelme: a database and web-based tool for image annotation. International journal of computer vision 77, pp. 157–173.

[79] Lin T.Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollár P. & Zitnick C.L. (2014) Microsoft coco: Common objects in context. In: European conference on computer vision, Springer, pp. 740–755.

[80] Hariharan B., Arbeláez P., Bourdev L., Maji S. & Malik J. (2011) Semantic contours from inverse detectors. In: 2011 International Conference on Computer Vision, IEEE, pp. 991–998.

[81] Liu C., Yuen J. & Torralba A. (2011) Nonparametric scene parsing via label transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, pp. 2368–2382.

[82] Mottaghi R., Chen X., Liu X., Cho N.G., Lee S.W., Fidler S., Urtasun R. & Yuille A. (2014) The role of context for object detection and semantic segmentation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 891–898.

[83] Bell S., Upchurch P., Snavely N. & Bala K. (2013) Opensurfaces: A richly annotated catalog of surface appearance. ACM Transactions on graphics (TOG) 32, pp. 1–17.

[84] Vondrick C., Patterson D. & Ramanan D. (2013) Efficiently scaling up crowdsourced video annotation. International journal of computer vision 101, pp. 184–204.

[85] Doermann D. & Mihalcik D. (2000) Tools and techniques for video performance evaluation. In: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, vol. 4, IEEE, vol. 4, pp. 167–170.

[86] Dutta A., Gupta A. & Zissermann A. (2016) Vgg image annotator (via). URL: http://www. robots. ox. ac. uk/˜ vgg/software/via .

[87] (2018) "labelbox: a collaborative training data software to create and manage labeled data for computer vision applications" available: https://labelbox.com/ .

[88] Maaten L.v.d. & Hinton G. (2008) Visualizing data using t-sne. Journal of machine learning research 9, pp. 2579–2605.