TECHNISCHE UNIVERSITÄT
CHEMNITZ

Image Based Object Detection and Tracking

Master Thesis

Faculty of Electrical Engineering and Information Technology
Chair of Communications Engineering

Submitted by
Mainuddin Ahmed
February 06, 2020

Supervisors
Univ.-Prof. Dr. Klaus Mößner
Chair of Communications Engineering
Technische Universität Chemnitz

Priv.-Doz. Dr.-Ing.habil. Dirk Joachim Lehmann
Data Analytics Powertrain Mechatronics
IAV GmbH

Abstract

The necessity of real-time object detection and tracking is growing and it has seen some significant improvement in recent years due to the development of deep learning. However, detecting the 3D location of an object from a 2D image is a complex task, since information is lost while mapping a 3D object to a 2D image plane. Therefore, for 3D reconstruction, the camera needs to be calibrated to obtain camera characteristics and the location, along with the orientation of the camera with respect to the real world. Besides, tracking an object in real-time, from frame to frame is also a challenging task because of the change of information from frame to frame and the irregularity and varying shape of the object and several other aspects. Various methods have been studied for object detection and tracking, most of which require multiple image information and they are computationally expensive.

In this thesis work, an object detection and tracking algorithm has been proposed, which uses a deep neural network architecture developed by He et al. for object detection and it tracks the detected object based on a single tracking point, which is also used for the 3D localization purpose. A linear interpolation based back-projection method has been developed, to project the tracking point into 3D. Hence, this method does not require any camera calibration and only one image is required for detection and tracking. However, it works when the object is moving on a flat surface and for the back-projection, the object needs to be in a known environment at the beginning.

Keywords: camera calibration, object detection, tracking, stereo vision

Acknowledgements

Firstly, I would like to show my gratitude towards The Almighty God without whose help this task would not be possible. My appreciation and thanks go to my honorable supervisor Prof. Klaus Mößner from TU Chemnitz for showing keen interest in my task and providing his valuable supervision. I am greatly indebted to Prof. Dirk J. Lehmann, my supervisor from IAV GmbH. Working under Dirk was an incredible journey of learning for me. He did not just teach me the art of approaching a problem from a scientific perspective and breaking it into smaller tasks but also gave me motivation and inspiration whenever I was getting stacked.

Besides, I would also like to thank my colleagues from IAV GmbH, especially the Vision team and my friends from TU Chemnitz. These people have no idea how much I have learned from them.

Finally, I have to mention my family who has supported me throughout my whole life and helped me to come where I have come today.

Contents

1 Introduction

This chapter discusses the motivation behind this work, the necessity of a simple and fast tracking method, followed by the goal and expectation of the thesis and it ends with a short description of the organization of the rest of the report.

1.1 Motivation

Object detection and tracking has a continuously growing application in industrial robotics, video surveillance, traffic monitoring, pedestrian detection, human computer interaction and so on. [1]

Object detection can be described as the combination of two separate tasks - image classification and object localization, where image classification is to recognize the classes of the objects present in the image and object localization is to identify the locations of the objects.[2]. The breakthrough of object detection method can be marked by the availability of large and diverse data sets like ImageNet in 2010 [2] and the successful usage of deep convolutional neural networks(CNN) by Krizhevsky et al. [3] in 2012 for image classification. Since then the deep learning research community has seen several algorithms which have shown significant advancements.

Object tracking is the method by which the object of interest is localized in each frame of the video. It can be a complex problem considering the challenges of complex movement and irregular shapes of the object and background noise etc.

Moreover, when it comes to 3D object detection and tracking from 2D image, obtaining 3D information from 2D image is a challenging task since from 3D to 2D transition depth information is lost. For that, camera parameters need to be known and the process of finding the camera parameters to obtain relationship of the image of the camera with the 3D world is called camera calibration. Various camera calibration method have been studied in last decades. However, it is a complex process. A simpler alternative is single image based back-projection method.

The usage of object detection and tracking has been increasing in our daily lives, as well as in the industry, which demands for simpler, faster yet accurate and reliable object detection and tracking algorithm.

1.2 Objectives

The goal of this thesis is to develop an algorithm to detect and track a single object in 3D based on a single image. The whole task can be subdivided into several key

modules. The object needs to be detected and classified and then a single tracking point needs to achieve as well. Since this object is going to be represented by only this single tracking point, therefore it is important to make sure that the algorithm detects such a tracking point which is always on the body of the object, even if the object has a very irregular shape. The next step would be to reconstruct the tracking point in 3D. Therefore, a 2D to 3D back projection algorithm has to be developed. However, since it is a single image based system, information about the 2D and 3D position of the object need to be retrieved from the environment. For that purpose, a reliable and robust convolutional neural network based handwritten coordinate detection module need to be developed.

A Graphical User Interface(GUI) will be developed for the user, which provides options for choosing different mode of operations and to control the operations.

1.3 Overview

The organization of the report is as follows: in **chapter 2**, necessary background about CNN, Mask-RCNN, camera calibration and skeletonization has been given, which is followed by the discussion about the state of the art methods for stereo and monocular camera calibration methods. **Chapter 3** presents the state of the art researches in the domain of object detection, object tracking and optical character recognition. **Chapter 4** presents the approaches for implementing the solution and **Chapter 4** discusses the result of experiments and evaluate the proposed method. At the end, **chapter 5** contains the summary of the whole work, its limitation and future possibilities.

2 Background

In this part, the necessary basics of the core concepts used in this work have been explained in short. It starts with the model representation of a camera, then gives a high-level idea of the building blocks of a convolutional neural network and their functionalities. The last section is a discussion on Mask R-CNN, YOLO and SSD object detection model, the main model used for the framework.

2.1 Camera Calibration

The history of the camera starts at around 470 to 391 BC by the Chinese philosopher Mozi, who proposed the idea of light traveling straight from the object to the camera through a small hole and creating an inverted image. This phenomenon is known as camera obscura, which is a Latin term meaning dark room. However, Arab physicist Ibn al-Haytham in the 11th century first demonstrated this through experiment [4], which was the invention of the pinhole camera and this model is known as the pinhole camera model.

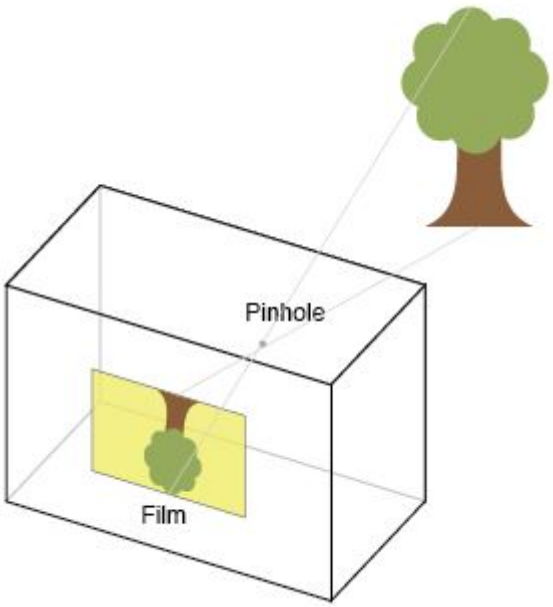


Figure 2.1: Pinhole camera model. Source: [5]

The pinhole camera model is the simplest camera model that describes how a

3D scene is mapped on a 2D plane, fig.: 2.2. The image which is generated inside the camera is a 2D map or projection of the 3D scene of the outside world. This projection is called perspective projection.

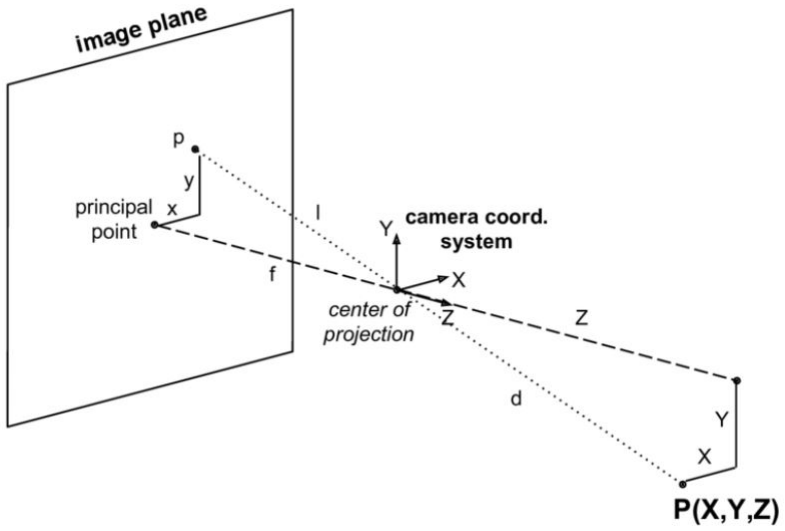
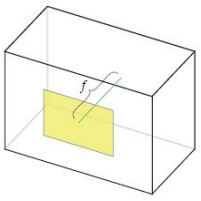
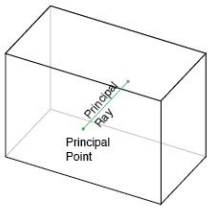


Figure 2.2: Here, in this image P(X, Y, Z) is the position of the object in the 3D world, which is the world coordinate. Camera Coordinate is the position of the camera in the 3D world, which is also the center of projection. Reflected light travels from the object or world coordinate to the camera lens or camera coordinate and then it is projected onto the 2D image plane, which is here p(x, y). Image Source: [6]

To understand how the 3D world is related to the 2D image, the internal and external parameters of the camera needs to be known, where the internal parameters are focal length, distortion, skew and image center of the camera and external parameters are the position and angle of the camera in 3D real-world [7]. The method to estimate these parameters is known as camera calibration.



(a) Focal length



(b) Principal point.

Pinhole camera model follows projective transformation from R3 space to R2 space mapping which can be described by the following equation:

$$P' = K[RT]P$$

where, P' is the 2D projection of 3D point P.

K is the camera matrix or the relationship between lens and sensors. It consists of camera center $C(C_x, C_y)$, which is the position of the pinhole from where the light enters and focal length $f(f_x, f_y)$ or the distance between the pinhole and the image plane.

R is the rotation matrix, describes the spatial angular position of the camera with respect to the world.

T is the translation matrix, which describes the spatial distance of the camera from the object.

Thus finding the camera matrix K gives the internal camera characteristics, on the other hand, from R and T external camera parameters can be found.

The way to calibrate a camera is to take a picture of a known object whose 3D location P and if the internal parameters are known then the projected point can be calculated. However, internal parameters may not be available, so in that case, they have to be calculated by solving the above mentioned projective transformation equation.

The most common method as mentioned earlier is the Zhang method, where a checkerboard whose formation is known is used for this purpose and multiple or at least two image needs to be taken to get enough information to solve the equation. That is how the 3D point is projected on a 2D plane.

2.2 Back Projection

While forward projection is 3D to 2D projection, backward projection is the other way - 2D to 3D projection. In forward projection, 3D world coordinate is transformed to image coordinate through 2D camera coordinate. In backward projection, 3D world coordinates are retrieved from 2D image coordinates using the world to camera relationship. The process has been illustrated in figure 2.4 and in 2.5.

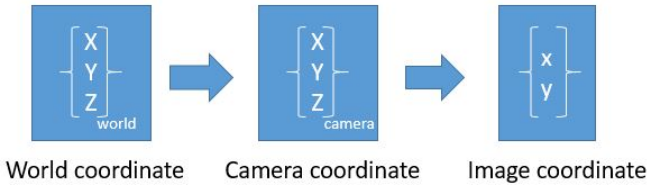


Figure 2.4: Forward projection

Camera parameters can be achieved from the forward projection model, which gives the relationship between the 3D world where the object is and the 2D world of the camera. Using these parameters, a 2D image plane can be transformed into a 3D plane or any point in the image can be reconstructed in 3D.

As discussed in the previous section, to retrieve those camera parameters usually multiple images are used because they provide the necessary information. However, there are back-projection methods which do not need camera calibration to gain

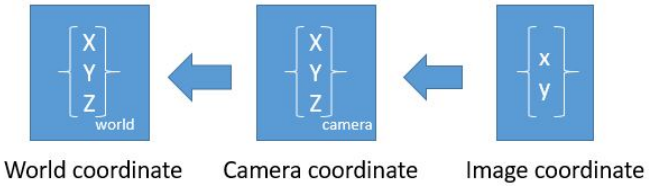


Figure 2.5: Backward projection

camera parameters if they can get some sort of geometrical information about the object or the 3D environment of the object.

2.3 CNN

One of the crucial tasks of the proposed back-projection algorithm is to detect handwritten coordinates. The quality of the detection decides the accuracy of the back-projection. Therefore, it is very important to develop a reliable handwritten number detector.

Human handwritings are not similar to each other. In fact writing of every single human is unique. This uniqueness together with the background noise and other challenges makes character recognition one of the most challenging classification tasks. Traditional classifiers are not very good at handling a wide variety of cases. On the other hand, the main strength of CNN(Convolutional Neural Network) based classifier is that, based on the quality and the quantity of the data of the training set they are very robust against variation. In some cases, they even reached close to human-level performance. Another advantage of CNN is that the algorithms are general, so the same algorithm can be trained on a different dataset and then they can be used for detecting different objects. For these reasons, for the handwritten coordinate detection module, we have chosen to develop a CNN based multi-digit number detector. In the following section, an overview of CNN and its basic components has been discussed.

Convolutional Neural Network or CNN is a multi-stage based neural network architecture which is used for processing information. As its name suggests, in CNN convolutional operation is used at least once in the process. In recent years, CNN has achieved remarkable improvement. Although the way the human brain processes visual information, was proposed by Hubel et al.[8] in 1962, the first modern development of CNN was Yan LeCun's LeNet[9] in 1998.

Since then, lots of remarkable architecture have been developed but there are four basic operations that almost all CNN possess. They are

1. Convolution
2. Activation function
3. Pooling
4. Fully Connected Layers

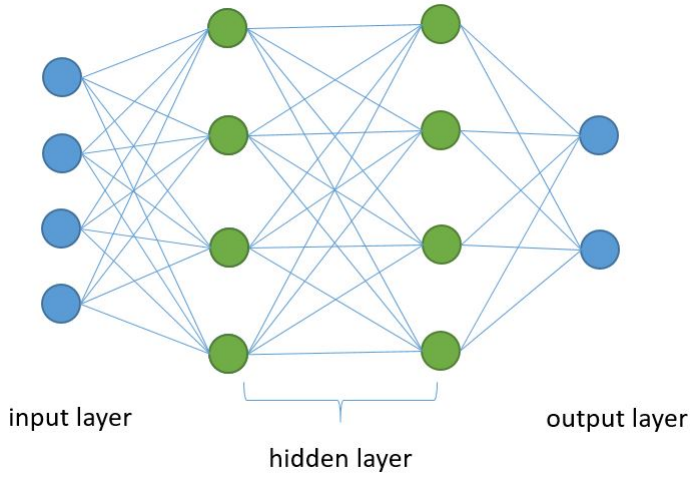


Figure 2.6: Neural Network

2.3.1 Convolution

Convolution is the mathematical operation between two functions, where the output function is a specific feature of one of the input functions shaped by the other one. It can be defined by the following equation:

$$(i * k)(t) = \int_{-\infty}^{\infty} i(\tau)k(t-\tau)d\tau \tag{2.1}$$

In the context of a convolutional neural network, function i is the input image, function k is kernel or filter and the output is a feature map. Here, i and k can be multi-dimensional arrays, where the number of the channel of the input image determines the number of dimensions and the dimension of k depends on the number of kernels or filters being applied on the input image.

The purpose of applying convolution on an image is to extract features or important characteristics from the original image while maintaining the spatial relationship among the pixels. For that, kernels or filters are used. Different kernel matrix can extract different types of features depending on their combination of values. For example, convolving an edge detector filter with an image can extract only the edges, a blur filter can remove the details. Besides, the usage of convolution helps to reduce information processing and increases efficiency.

In classical image processing, hand-crafted kernels are used but they can cover only certain types of scenarios. On the other hand, in a neural network, the kernels are readjusted through the feedback of the learning algorithm and because of that, the network can learn a wide range of features about an object. This is why the performance of a CNN greatly depends on the number of images the network has been trained on and that also brings a limitation, since processing a large number of images requires processing power and memory. However, the development of GPU(Graphical Processing Unit) makes it possible to train and test a complex

neural network with large data set.

The size of the output feature map depends on the three factors:
i) Depth: The number of filter,
ii) Stride: The number of pixels the filter will move during convolution and
iii) Zero padding: Adding zeroes around the image matrix symmetrically. Helps to preserve the size of the input image.

2.3.2 Activation Layer

An activation function is the method of mapping input to output based on a threshold and the activation layer is used right after convolution operation. The simplest activation function is a linear activation. However, as the real-world data is not linear, non-linear activation functions are used to introduce non-linearity, because through linear mapping it is not possible to learn complex functions or features. The dimension of the image remains the same after the activation layer.

The most widely used traditional activation functions were hyperbolic tangent function and sigmoid function [10], but both of them suffer from saturation. Both of the functions saturate for high positive and negative values and they are sensitive for a very short range of values[11]. They are also affected by vanishing gradient descent problem when trained for a very deep neural network[11].

This problem is handled by Rectified Linear Activation Unit or ReLU [12] and according to research and experimentation this is the most effective activation function [13]. Since they are piecewise linear function they can resolve the issue of low sensitivity and high saturation. There are also some variations and improvement of ReLU, such as leaky ReLU or LReLU [14], parametric ReLU or PReLU [15].

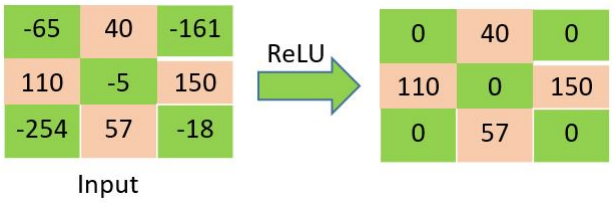


Figure 2.7: ReLU activation function operation

2.3.3 Pooling

The pooling layer is used to reduce the spatial input size and to extract dominant features by replacing the input values. The most common is average pooling and max pooling, where average pooling replaces a rectangular space of the input by its average, max-pooling takes the maximum value. Thus, average pooling can help filter noise and average pooling can reduce dimensionality. Pooling can also reduce over-fitting and makes the learning of the network translation invariant, which makes the CNN classifier very robust.

Figure 2.8: Example of max-pooling

It is common practice to use max-pooling in the middle of the network and average pooling at the final layer of the network as in ResNet and GoogleNet architecture. The output size of the pooling depends on the pool size or the spatial extent and the stride. Generally, stride size S = 1 or 2 is recommended.

However, Springenberg et al.[16] recommended to not to use pooling layer at all, instead of using bigger convolutional stride for reducing the spatial size nowadays, it is becoming popular to use lesser pooling layer for deeper network as in [17] by He et al., since their usage are showing more promising results. It is assumed that in the future there will be no use of pooling layer at all [18]. The pooling layer is used to reduce the spatial input size and to extract dominant features by replacing the input values. The most common is average pooling and max pooling, where average pooling replaces a rectangular space of the input by its average, max-pooling takes the maximum value. Thus, average pooling can help filter noise and average pooling can reduce dimensionality. Pooling can also reduce over-fitting and makes the learning of the network translation invariant, which makes the CNN classifier very robust.

2.3.4 Fully Connected Layer

Fully Connected Layer or FC comes at the end of the network. At this point, the output matrix of the previous layer is flattened and they go through a feed-forward network, which is a fully connected network of neurons. During this time the network learns about the low-level features and afterward the output of the feed-forward network is classified by a softmax classifier.

An example of a simple CNN architecture can be LeNet [9] in fig. 2.9. architecture, which has only 3 convolutional layers.

Research has proven that the learning of the network becomes more efficient and powerful with the increase of the depth of the network. Therefore, in the last few years, several deep, complex and improved neural network architecture have been developed which are very deep and they have achieved significant-high learning accuracy.

For instance, GoogLeNet (2014) [19] had 22 convolutional layers, where VGG Net (2014) [20] had 16 and ResNet (2015) [21] had 34 layers. In fig. 2.10 a comparison can be seen among the most successful architectures.

These models have been trained on a large dataset and they can be used as a

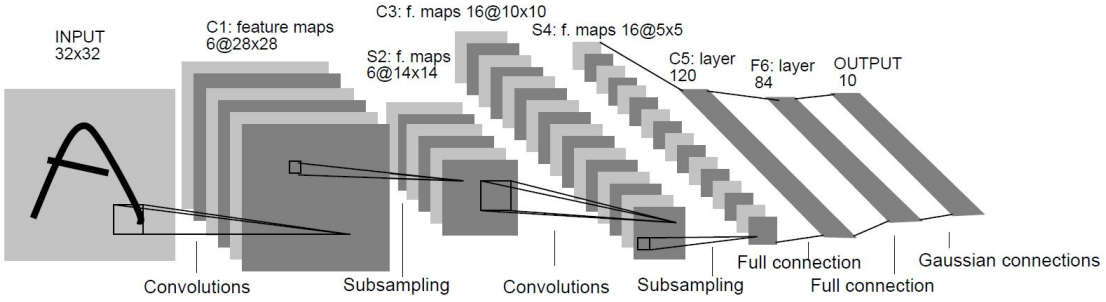


Figure 2.9: LeNet architecture. Source : [9]

Model	Size (M)	Top-1/top-5 error (%)	# layers	Model description
AlexNet	238	41.00/18.00	8	5 conv + 3 fc layers
VGG-16	540	28.07/9.33	16	13 conv + 3 fc layers
VGG-19	560	27.30/9.00	19	16 conv + 3 fc layers
GoogleNet	40	29.81/10.04	22	21 conv + 1 fc layers
ResNet-50	100	22.85/6.71	50	49 conv + 1 fc layers
ResNet-152	235	21.43/3.57	152	151 conv + 1 fc layers

"conv" and "fc" indicates convolutional and fully-connected layers, respectively.

Figure 2.10: Comparison of accuracy, number of convolutional layers among different leading deep CNN architectures. [22]

pre-trained model, which means their learned weight can be used to train a new dataset. New datasets can be trained on top of them Therefore, the new dataset does not have to be very large and it saves design time.

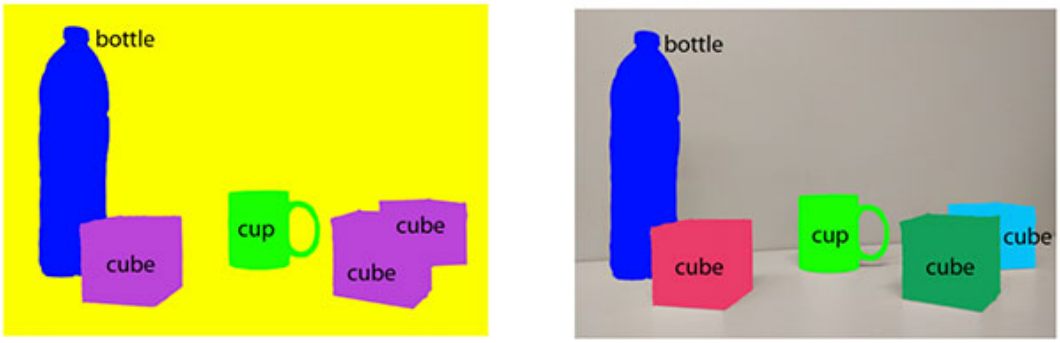
Such models have also been used as the base model for developing very powerful and fast real-time object detection models. The three most notables models are: R-CNN [23] family by Girschik et al., SSD [24] by Liu et al. and YOLO [25], YOLO9000 [26] and YOLOV3 [27].

2.4 Mask R-CNN

As explained earlier, object detection is the traditional computer vision problem of detecting the class of the objects present in the image and its location in the image as well. Therefore, the output of any object detection model would be a bounding box surrounding the object and the class label.

On the other hand, semantic segmentation is to classify each pixel of an image to a category, without distinguishing among different instances of a class. The next step is instance segmentation which performs both object detection and semantic segmentation.

Instance segmentation is particularly helpful in our case, where we have to detect the object and find out a tracking point on its body. Therefore, for our object



(a) Semantic segmentation. Source: [28]

(b) Instance segmentation. Source: [28]

detection and tracking purpose, we have chosen Mask R-CNN[29], which is an object instance segmentation framework. Along with the bounding box and class label, it also provides a binary mask of the object.

Mask R-CNN, developed by Girschik et al. is currently the most sophisticated deep learning model which has achieved state of the art result on the COCO data-set [30] and outperformed all the leading frameworks. It is a progressively developed version of the R-CNN(Regional-Convolutional Neural Network) family of CNN. From fig.: 2.12 it can be seen that the mask and bounding box prediction of Mask R-CNN is very accurate, where FCIS [31] with more complex architecture and winner of COCO 2015 and 2016, has artifacts.

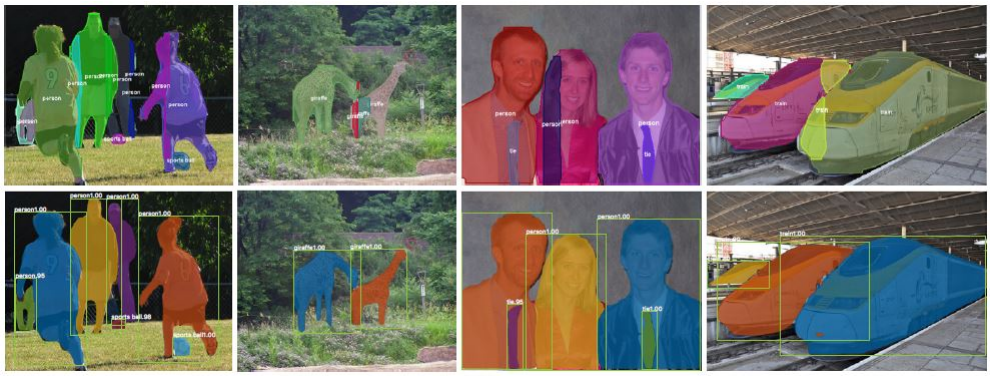


Figure 2.12: Comparison of performance of object detection by Mask R-CNN. FCIS [31] vs Mask R-CNN. The mask and object instance detection of Mask R-CNN is more accurate than more complex FCIS architecture. Source: [29]

Mask R-CNN Architecture

In the traditional classification method, the whole input image goes through CNN, where sliding window approach is used from the convolution to the max-pooling

phase, which is computationally expensive. On the other hand, in R-CNN [23] at first, a selective search algorithm is applied to find out some candidate object regions or RoIs and only these RoIs go through SVM classification.

Fast R-CNN [32] introduces RoIPool to extract a fixed-size feature map from every RoI. Based on these feature maps bounding box and class labels are detected.

In Faster R-CNN [33] Region Proposal Network is used rather than a selective search algorithm to directly propose candidate regions. After that likewise Fast R-CNN, RoIPool is used for feature extraction and from there bounding box detection and classification are performed.

Mask R-CNN is built upon Faster R-CNN where in parallel to the classification and bounding box detection branch, it also has a branch for mask prediction from each RoI. Nonetheless, for an accurate mask prediction, it incorporates some improvement to the existing RoIPool method.

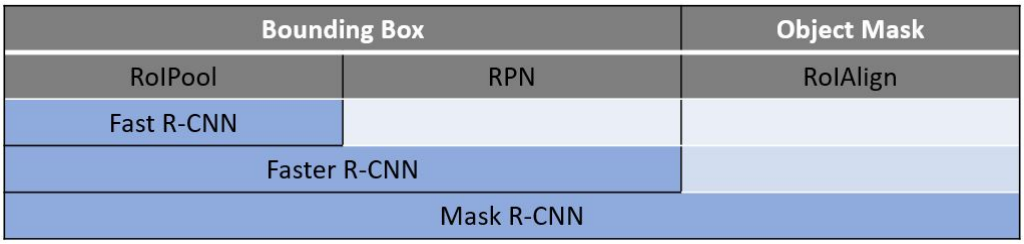


Figure 2.13: The architecture of Mask R-CNN. Mask R-CNN is developed on top of Faster R-CNN, which uses RPN and RoIPool for bounding box detection. Again, Faster R-CNN is built on top of R-CNN which introduced RoIPooling for feature extraction. However, Mask R-CNN incorporated RoIAlign to solve the quantization issue due to RoIPooling, and hence produces robust bounding-box mask detection. [29]

During the RoIPool process for feature extraction from feature maps, the selected RoIs go through two levels of quantization. Because of these quantizations, misalignment occurs between RoIs and obtained features. Therefore, Mask R-CNN addresses RoIAlign method to resolve this issue which is crucial for correct mask generation from RoI.

You Only Look Once : Unified, Real-Time Object Detection (YOLO)

Although we have chosen Mask R-CNN as the primary object detection model, it has some limitations regarding speed and processing power. It is not possible for real-time detection with Mask R-CNN. With GPU it can process in semi real-time but with CPU, we cannot do that. In our application, for detection with Mask R-CNN, the user has to give the command for detection and that detection is only for

one single frame. That is why we have also considered YOLO and SSD for real-time operation. Our application has the option of detecting objects with YOLO as well.

YOLO is one of the most powerful object detection model family which started from YOLO or You Only Look Once: Undefined, Real-Time Object Detection System [25] by Redmon et al. The significant improvement which YOLO introduced to the world of object detection family was speed. Compared to its contemporary architecture Fast R-CNN it was less accurate but it was the faster of that time - 45 frames per second processing time, although they also developed a shorter version, named Fast YOLO with 155 frames per second rate. However, Redmon et al. have presented two more upgrades of YOLO, where they improved the accuracy as well. Since YOLO is the base for the latest two, we will shortly discuss the architecture.

The main strength in YOLO lies in the fact that the authors have approached the object detection problem as a regression problem. Here, a single neural network is used to predict bounding and class probabilities and this happens in one pass, figure 2.14. This makes the algorithm fast since the network "sees" the image only once, no sliding window mechanism is required, hence the name You Only Look Once. There is also no complex pipeline like its counterpart R-CNN.

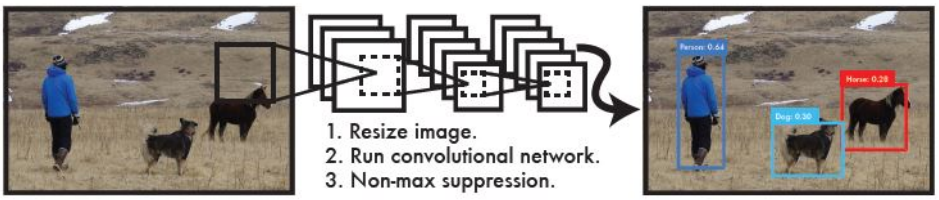


Figure 2.14: YOLO processing system. Only a single neural network is used only once to evaluate the image. [25]

Figure 2.15 explains the mechanism. The input image is divided into a 7x7 grid and 2 bounding box. Each of these bounding boxes has box locations and a box confidence score. From there a class probability map has been created. One interesting fact is that the network also learns about the surrounding context of each class and their relationship as well. YOLO is trained on the PASCAL VOC dataset.

However, YOLO suffers from accuracy issues when compared to its counterparts. It also does not perform well when the objects are too close to each other. In the later version - YOLO9000: Better, Faster and Stronger [26], the authors have emphasized more accuracy while maintaining the speed. In this version, the network has been trained on 9000 object classes.

The frame rate has been increased to 67 fps with 76.8 mAP, while the previous version had 52.7%mAP. Instead of a 7x7 feature map in YOLO, the second version has a 13x13 feature map. The model can also run on different sizes, which gives the opportunity to choose between speed and accuracy. Many improvements have been introduced here. For example, multi-scale training, convolutional layer based on anchor boxes which come from the feature extractor, dimension clusters, high-resolution classifiers and so on. As shown in figure 2.17, the anchor in the black

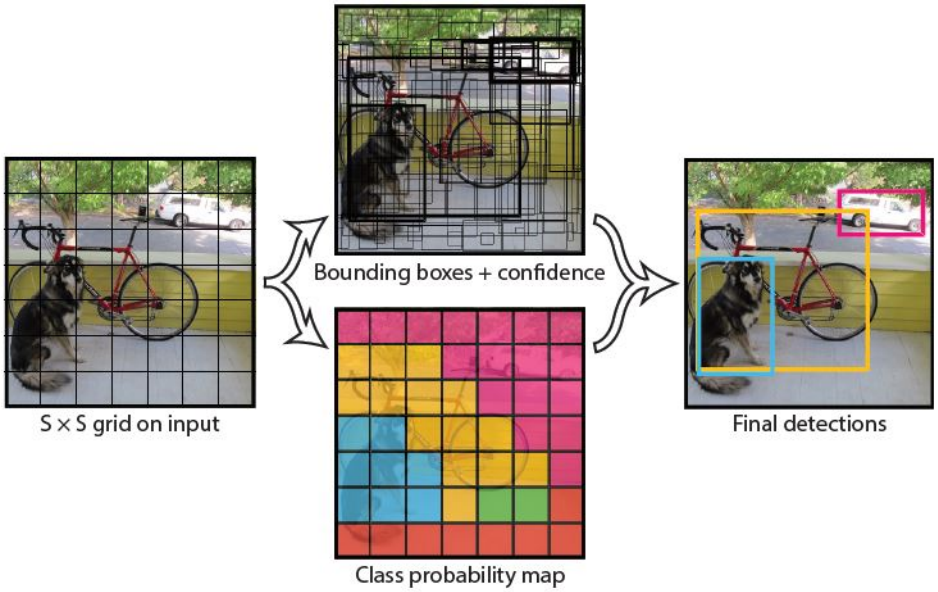


Figure 2.15: YOLO model. [25]

dotted box helps to decide the bounding box region in the blue line.

YOLOv3 is the latest in the family with an accuracy of 28.2 mAP, which runs in 22 ms, so this is three times faster than its strong competitor SSD. The second version of YOLO was faster than SSD but its performance was not better. However, in this version, they have been able to achieve that. Figure 2.18 shows how YOLOv3 is better than the state of the art models in the speed and accuracy trade-off. They have maintained the usual characteristics from the previous architectures, such as multi-scale training, batch-normalization and data-augmentation.

The core of the YOLOv3 architecture is the Darknet-53 which is an improvement to the Darknet-19 used in version 2. As its name suggests, it has 53 convolutional layers and their arrangement is such every 3x3 layer is followed by a 1x1 layer.

SSD: Single Shot Multibox Detector

"SSD: Single Shot Multibox Detector" [24] has been proposed by Liu et al. in 2016 and became a landmark in the object detection domain. It has an astounding accuracy of 74% mAP at 59 frames per second. SSD has been built upon VGG [20] network with some modification to the original VGG network, see figure 2.20.

As its name describes it has three main components, they are single shot and multibox mechanism. Single Shot means the mode the model performs object detection and localization in one single forward pass. Multibox is a strategy developed earlier by Szegedy which consists of confidence loss and location loss.

SSD has been trained on COCO and PASCAL VOC datasets. During training hard negative mining, data augmentation and non-maximum suppression have been

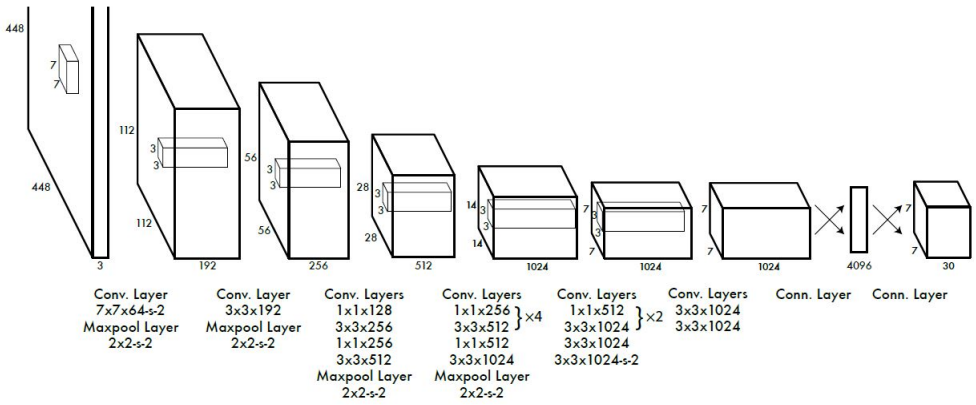


Figure 2.16: YOLO architecture. [25]

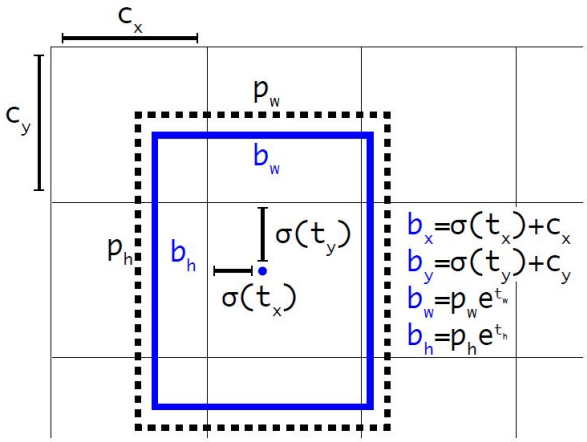


Figure 2.17: Dimension prior used in YOLO v2. [26]

used.

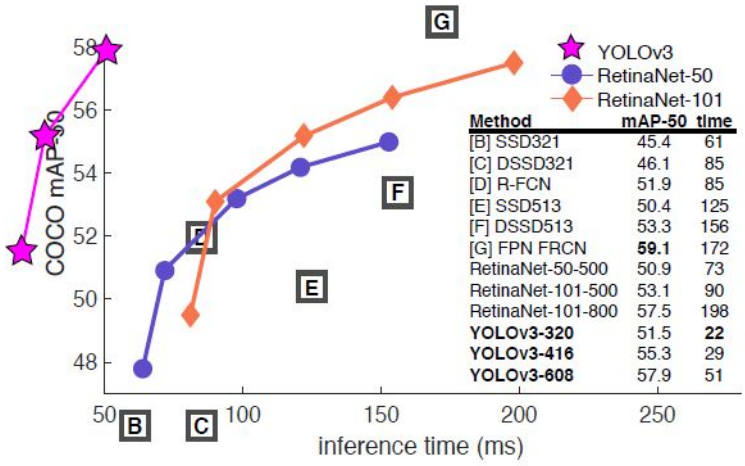


Figure 2.18: Performance comparison of YOLOv3 against other state-of-the-art achitectures. [27]

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2.19: Darknet-53 architecture. [27]

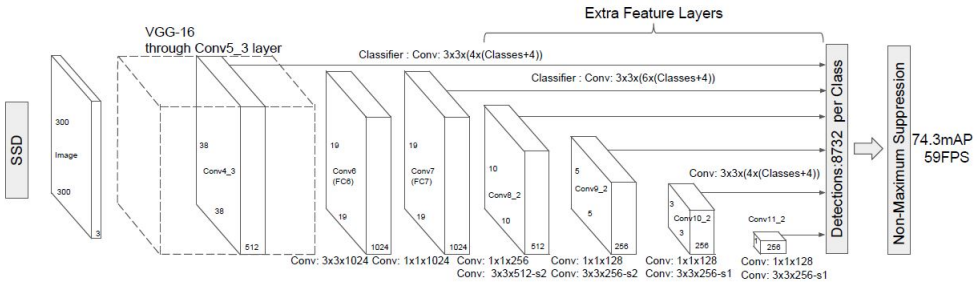


Figure 2.20: SSD architecture. [24]

3 State of the Art

The main components of this thesis are - object detection in 2D, its 3D reconstruction and then tracking. For object reconstruction, 2D to 3D back-projection has been used, which requires hand-written number detection. Therefore in this chapter, section 3.1 discusses different methods of 3D object reconstruction, section 3.2 about object tracking and section 3.3 about handwritten number detection.

3.1 3D object reconstruction

3D Object reconstruction can be divided into two branches [34]: modern deep learning based method and traditional method.

3.1.1 Deep learning based methods

Human beings are very good at recognizing objects. Even when they see only a part of the object, they can still visualize the rest. That is because we humans use our previous knowledge of the object. For the last few years, deep learning is also being used to apply the same principle for 3D object reconstruction. Since here, there is the advantage of training the convolutional network with large and varied data-set, which together with various advanced deep learning models, has shown quite promising results in this short period.

In deep learning based methods, the algorithm tries to learn the geometrical and structural information from one or multiple images. Both supervised and unsupervised learning have been tried for this purpose. For instance, Gadelha et al. have used PrGAN (Projective General Adversarial Network) in [35] to generate 3D shapes from 2D images. A GAN has been trained by multiple unknown two dimensional of multiple objects. At the heart of this method, there is a projection module. Once a 3D shape is created by the generator, the projection module creates a render and the adversarial network segregates the real images. The trained network can reconstruct 3D shapes from multiple images or given a single image it can predict the depth.

Choy et al. have developed a unique RNN(Recurrent Neural Network) called 3D-R2N2 [36], which uses both single and multiple images to reconstruct the object in the form of a 3D occupancy grid. Their proposed method can reconstruct a 3D shape of an object in voxel grid form, both from single or multiple images. The architecture is consists of a 2D-CNN which works as an encoder, followed by a 3D LSTM which tries to formulate pattern based on previous observations and then 3D-DCNN (Deconvolutional Neural Network), which works as a decoder.

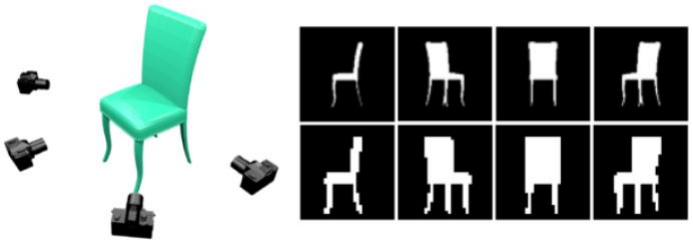


Figure 3.1: Images are taken from eight viewpoints of an object. Image source: [35]

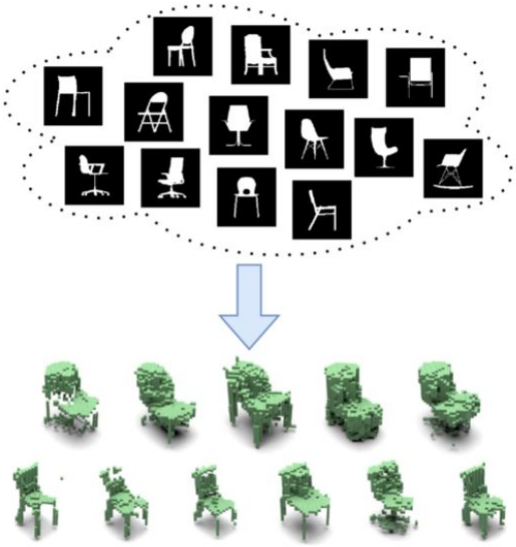


Figure 3.2: 3D shape generation, given a number of 2D images. Image source: [35]

The advantage of the CNN based methods is that they do not need to know camera parameters. However, they are expensive in terms of computational time and memory.

3.1.2 Traditional methods

The goal of the traditional calibration or reconstruction methods is to retrieve a mathematical algorithm from the geometrical relationship between the 3D world and a 2D image, which is used for calibration and 3D reconstruction. There are both single image based and multiple image based methods but since the proposed method of this thesis is a single image based on, more focused is given on these types of works.

In the past decades, researches have been conducted in the area of the calibration method. They are both multi-image based and single image based. Where in multi-image based method, there are stereo vision[37], which uses several images from a

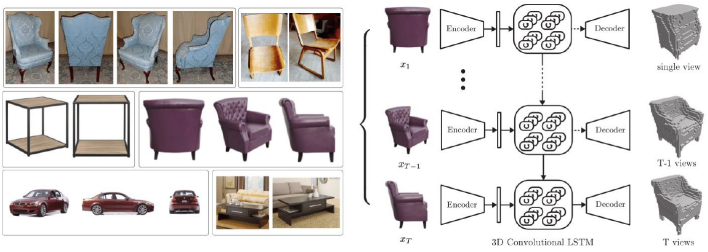


Figure 3.3: 3D-R2N2: Learns from the image of the object from different viewpoint. Image source: [36]

different angle of view and after that depth information is retrieved from them with the help of the triangulation method. Some other methods have been discussed below.

Shape-from-silhouette

In this popular method for 3D reconstruction, multiple silhouette images from a calibrated camera are taken and then the shape of the image is approximated, where often the output is a visual hull [38]. According to Cheung et al. this method cannot produce a good result if the number of silhouette images is less. In their experiment, a relatively better output could be found after using 66 images. Some related works regarding this method are : [39], [40], [41] and [42].

Single Image Based Method

Single image based methods do not have the advantage of extracting the geometrical relationship between the object and the world, which is the case in multiple image based methods. Therefore generally, the single image based methods demand to know some kind of 3D geometrical information beforehand for calibration purposes. Some works have been discussed below.

Wilczkowiak et al. have developed a method of 3D modelling and calibration using parallelepipedes in [43], [44] and [45].

Deutscher et al. [46] have used Manhattan World model (fig.:). The image scene satisfies the Manhattan world assumption where the image contains three orthogonal directions. However, they have stated that the calibration model is less accurate than the other methods but it is good for tracking purposes in an urban environment.

Park et al. have developed a Manhattan world based method [47] to reconstruct both outdoor and indoor scenarios. The main target application is augmented reality. In this algorithm, the image is segmented into several planes. After that cost functions are defined for optimizing graph cut optimization. Then the segments are grouped according to neighborhood and depth. Ranade and Ramalingan have concentrated on reconstructing the line segment using Manhattan world algorithm [48].

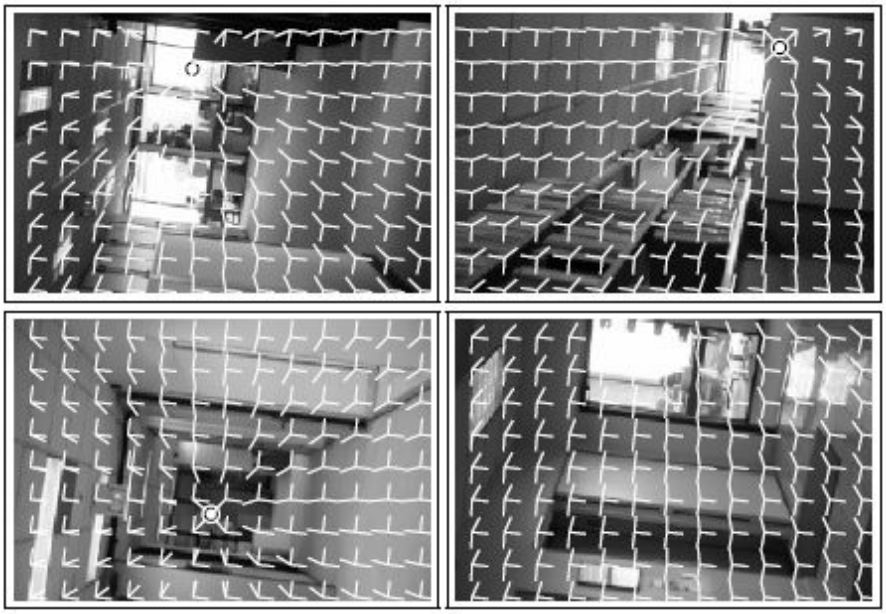


Figure 3.4: Camera calibration using Manhattan world image. Vanishing lines are being indicated by the white lines. Source: [46]

Guillue et al. have proposed a vanishing point-based method [49], which needs at least two Vanishing points in the image, a 3D line segment length, principal point and aspect ratio. Similarly, Wu et al. [50] have also used vanishing points and vanishing lines to obtain the intrinsic camera parameters but they have incorporated RANSAC for outlier elimination. The algorithm proposed by Miyagawa [51] is very much similar to ours, where they have utilized five known points on a 1D object, where at least three points have to be co-linear. Using these five points an equation is derived which gives the camera parameters.

From the above discussion on different methods for object reconstruction, it can be seen that the deep learning methods are better and more robust than the traditional methods. Nevertheless, they require more data and computation time, so they are not suitable for real-time applications.

On the other hand, among the traditional methods, the multiple image based or methods of reconstruction from shadow or motion recover the relationship between the world coordinate and the camera coordinate during the forward projection process.

Single image based methods are mostly comparable to our algorithm and here we can see a common pattern among them that they need help from the object scene environment to acquire geometrical information.

3.2 Object Tracking

Object tracking is a challenging task due to the change of position, lighting and sometimes even the shape of the object. Similar to object reconstruction, the problem of tracking has also been studied in different aspects.

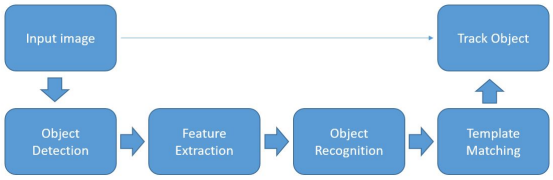


Figure 3.5: Tracking Pipeline [52]

Depending on the target applications there are different approaches to object tracking. These approaches can be divided into two: single frame based and temporal or multiple frame based. In single frame based methods, the object is detected by the object detection algorithm and then the object is tracked in every single frame using a tracking algorithm. Here the tracking process in each frame is independent of other frames. On the other hand, in the temporal methods, based on multiple frames the average position of the object is detected and then the tracker is updated from frame to frame. See [53], [54]. Besides, classical trackers, in recent times work has been done on pre-trained trackers but since the focus of this thesis is online tracking, therefore offline pre-trained tracking is out of the scope of this discussion.

According to Yilmaz et al. the classical methods can be classified into three categories:-

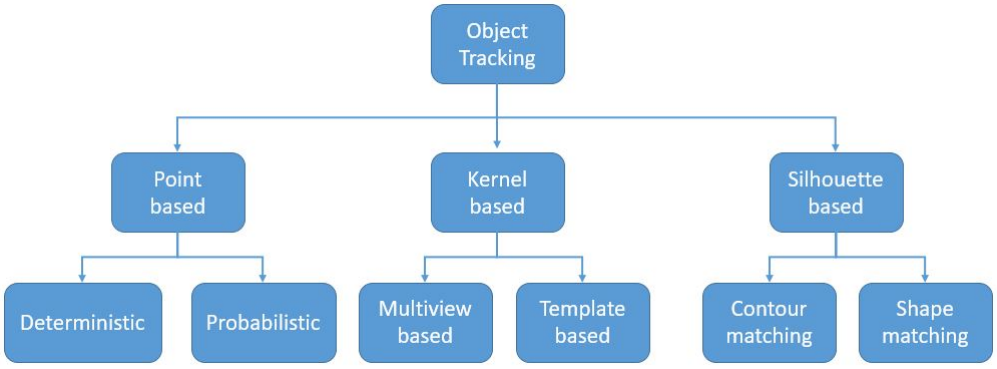


Figure 3.6: Classification of tracking methods [54]

Point based : In this methods, the object is represented by a single or multiple points. In the proposed method [55] of Kloihofer and Kampel, points of interest are generated from a region of the image. Once the interest points have been generated, SURF feature descriptors have used for tracking (fig.: 3.7).

Tissainayagam and Suter [56] have combined two Bayesian Multiple Hypothesis(MHT) algorithm, where in the first one, contours are segmented from edge map

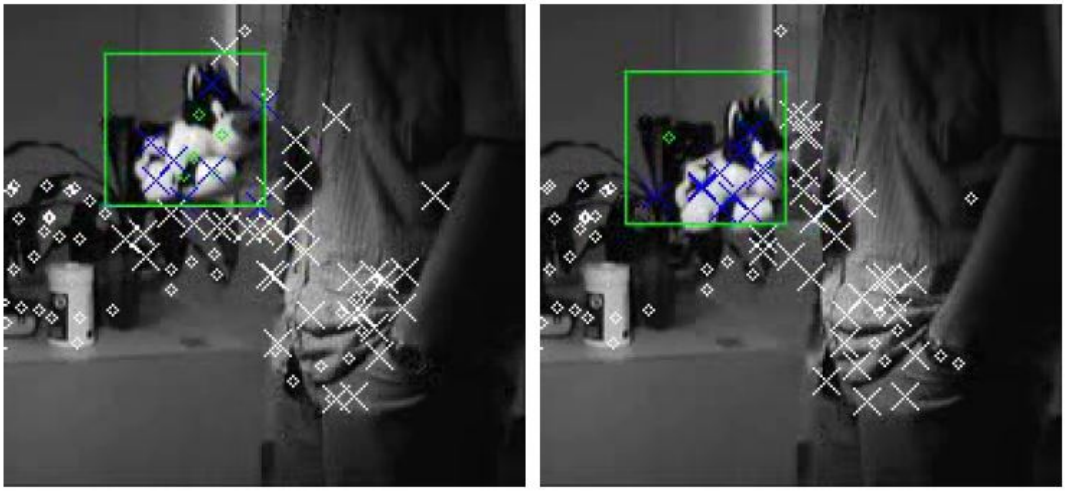


Figure 3.7: Interest point based tracking. [55]

and then the contours are merged together to form object shape. In the second algorithm, key points detected from the edge map are tracked, in combination with the object shape created from merged contours.

Zhu et al. [57] have extracted features from edges of the image and then the features were compared with the features of reference images, from sequence to sequence and hence tracked.

Serby, Meier and Gool have combined several low-level features - for instance, interest points, edges and color information, to develop a general object tracker. After the feature extraction, they have used particle filter for combining the features. However, in the initial phase, the user needs to select the ROI.

Kernel based : A geometric shape is used to refer the object. Used in multi-frame based temporal tracking methods. Such as, kernel based Mean Shift (MS) tracker [58], 3D tracking using multiple-camera [59].

Silhouette based : The shape of the object is used to track the object in each frame, generally through the contour or the skeleton of the object. This method is also dependent on previous frames. Some contour-based works are - [60] and [61], contour initialization from optical flow [62] and in [63] optical flow and edge detector have been applied.

In our work, our tracking method has been inspired by both the point-based and silhouette based methods, wherein silhouette based methods the shape of the object is achieved by various methods and then it is used for tracking. In our case since we want to track the object in 3D, so reconstructing the whole shape would be more complex than reconstructing just one single point. Hence, we will use only one point for tracking but unlike typical point-based methods, where the point comes from extracted features, we will combine the object detection approach to obtain the tracking point to ensure the reliability. In the proposed method, when the object

will be detected by a deep learning based highly accurate object detection model, the single tracking point will be extracted from the skeleton of the object.

3.3 Handwritten Number Detection

In this thesis, for backpropagation operation, the algorithm needs to read some handwritten 3D coordinates, which are decimal numbers. Numbers are a combination of digits positioned conventionally, therefore the core task here is to recognize handwritten digits. The handwriting of every human is unique which poses a great challenge in recognizing the numbers or letters correctly, hence the classical image processing algorithm was not very efficient in this domain. Successful results started to show while incorporating machine learning algorithms such as SVM (Support Vector Machine) and KNN (K-Nearest Neighbor). However, the biggest contribution to detect handwritten numbers was the MNIST data-set. In this section, we will discuss different data-sets and algorithms developed for handwritten digits.

3.3.1 Data set

The success of a neural network based classifier depends heavily on a good data-set. For a handwritten digit recognizer, the data set needs to have a collection of a wide range of variety of samples. The most notable of them are MNIST and SVHN.

MNIST : MNIST(Modified National Institute of Standards and Technology) [64] has become the standard data set for this digit detection purpose. It was developed by LeCun et al. in 1998 from the NIST dataset. It is a very rich collection of handwritten single digits with many different writing styles and they are prepossessed for the easiness of further use. It contains 60,000 images, where 50,000 are for training and the rest 10,000 are for testing. Each 28x28 pixel size image contains a single digit, which is normalized and centered to 20x20 pixel, maintaining the original aspect ratio. This data-set is labeled by 10 classes from 0 to 9.

Street View House Numbers(SVHN) : SVHN [65] is also a very robust data set created by google from its Google Street View images. It is a massive and diverse collection of 600,000 images, which is a very good resource for research and experiments. See fig.: 3.9

It has a similar preparation like MNIST of 10 class labels of 0 to 9 and each single number is resized to 32x32 pixel size.

3.3.2 Approaches

The application of neural networks in detecting handwritten digits is not new. The first attempt was in 1998 called MLP (Multi-Layer Perceptron) by LeCun et al. [64], although then the error rate was 4.7% but through the growth of the computing power and hence the increasing complexity of the network today's models are achieving close to 100% success rate. However, in the meantime, even till recently,

Figure 3.8: A sample of MNIST data-set images. Source: [64]

traditional machine learning algorithms such as KNN and SVM were also very popular. Here, the principal difference with the neural network models was that the machine learning models are used for classification after the features were extracted from the image. Where on the other hand, in CNN, feature extraction happens inside the network itself. Therefore, the proposed methods using those models have different feature extraction procedures. Another major difference is that most of the KNN and SVM methods require preprocessing. Here, some of the recent works have been discussed.

Machine learning based methods:

Babu et al. [66] have extracted four certain types of features, which are: i) the number of holes, ii) water reservoirs in four directions, iii) maximum profile distances in four directions and iv) fill-hole density. Then KNN has been used for classification. Before feature extraction noise reduction has been performed by the image pixel-wise.

Tuba and Bacanin [67] have used projection histogram with SVM. Their goal was to detect Persian and Arabic numeric. (In MNIST benchmark has achieved a good 99.05% accuracy.)

Gao et al. [68] have extracted some specific features from the images, the features are: Euler Number, roundness, moment feature, crossing density and pixel density. After that SVM has been used for classification.

Boukharouba et al. [69] have used Chain Code Histogram (CCH) for feature extraction and SVM for classification. With the help of CCH - to find the outline of the digit.

CNN based methods:

Islam et al. [70] have designed CNN with only one layer and have achieved a good score of 99.6%.

Goodfellow et al. (2013) [65] have developed a deep convolutional neural network to detect house numbers from the SVHN dataset. Instead of following three steps

Figure 3.9: Street View House Numbers(SVHN) [65]

of the traditional approach of multi-digit number detection, which are - i) localizing the numbers, ii) separating them and iii) recognizing them individually, they have followed an end to end pipeline of detecting a whole number.

They have found that the depth of the network plays a crucial role in increasing the accuracy of CNN. Their CNN consists of 8 convolution layer, followed by 1 locally connected and 2 densely connected hidden layers. This design achieved 96% accuracy on number detection and 98.7% accuracy on single-digit detection. Max pooling and normalization have been used.

The current CNN models for digit detection are inspired by the concept of increasing the depth of the network. Besides, the use of recently improved activation

Model	**Accuracy**	**Author**	**Year**
KNN	96.94%	Babu et al. [66]	2014
SVM	99.05%	Tuba and Bacanin [67]	2015
SVM	96.3%	Gao et al. [68]	2015
SVM	98.48%	Boukharouba et al. [69]	2017
CNN	95.3%	LeCun et al. [64]	1998
CNN	99.6%	Islam et al. [70]	2017

Table 3.1: Comparison of different methods of handwritten digit detection based on MNIST data-set.

functions such as : ReLU, adam, along with batch normalization, data augmentation and learning rate decay have shown very impressive results in the MNIST data set based Kaggle competition board.

Among the most available digit and number detection dataset SVHN certainly has a very rich collection than MNIST serves our purpose well and also it is a very clean and well-prepared dataset. Among the different approaches, the CNN based methods are showing clear distinctions than machine learning and traditional algorithms. The current researches and experiments are suggesting towards deeper and wider network and in the MNIST dataset several methods have been able to achieve 0.4% error rate. However, in order to achieve 99.7% accuracy we have used not just deeper network but also tuned carefully different hyperparameters.

Our algorithm requires to detect multi-digit numbers, not just digits. There are several deep learning based method, which can detect regions of the texts. One of these text detectors for example, "EAST: An Efficient and Accurate Scene Text Detector" [71] by Zhou et al. have shown good performance (fig.: 3.11). It has an end to end fully neural network-based pipeline (fig.: 3.10), so unlike other text detectors, it does not need sub-algorithms, which are computationally expensive. Text detectors are helpful when the layout of the text is unknown but since in our case the coordinates will be in one single rectangular box we have utilized this feature to detect the ROI through few preprocessing steps. That is why no deep learning based text detectors have been used, which makes it faster and also our method works even when the images have not been taken from an orthogonal position.

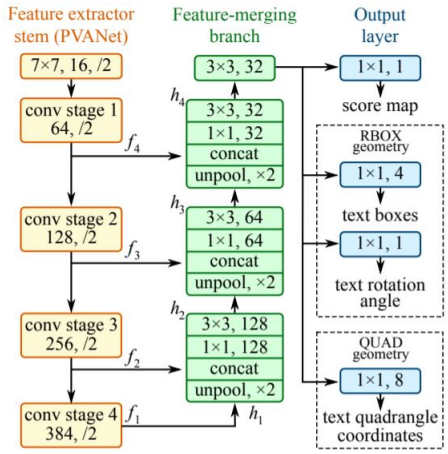


Figure 3.10: EAST text detector has an end-to-end neural network based architecture, so it does not require any computationally expensive sub-algorithms. [71]

We have seen researches from different perspectives on four topics: object reconstruction, object detection and tracking and lastly digit or number detection. In the proposed method we combination of all of these tasks. object tracking itself consists of object detection and since we are tracking it in 3D we also have to do object reconstruction. In our work, we have combined deep learning based object detection

Figure 3.11: Some sample text region detectino by EAST text detector. [71]

method, so that we can detect any object type with large variation with very high accuracy. For tracking, a point-based object tracking has been developed, which is fast and also reliable. For object reconstruction back projection is required, which needs number detection. For that, a CNN based digit detector has been developed but for text region or number detection, a unique method has been offered.

4 Implementation

In this chapter, it has been discussed how the proposed detection and tracking algorithm has been implemented. For the convenience of the reader, here is a short recapitulation of the whole functionality.

The objective of this thesis is to detect an object from a single 2D image and from there a tracking point is detected. To locate the tracking point in 3D, back-projection from 2D to 3D is performed. When the tracking point is found, then with the help of the provided positional information in the object environment the required 2D and 3D coordinates for the back projection are collected. Through the proposed back-projection algorithm, the 3D position of the 2D tracking point is reconstructed and tracked in the video from frame to frame.

Therefore, the architecture can be divided into four sub-components, they are:
i) Object detection,
ii) Finding a tracking point on the object,
iii) Detecting handwritten coordinates and markers to obtain the 2D and 3D locations of the object and
iv) Back-projection for 3D reconstruction of the tracking point

However, it is important to describe the test environment first. After that, the above-mentioned steps have been discussed in the following sub-sections.

4.1 Approach

In this section, the test environment has been introduced first. Then the back-projection algorithm has been explained mathematically.

4.1.1 Test Environment

As discussed in 3.1.2, since the geometrical relationship between the 2D image and 3D world cannot be deducted from a single image, therefore geometrical information needs to be known beforehand. In our proposed method, three known points are required around the object. Their 2D and 3D coordinates have to be extracted.

As seen in fig:4.1 any three arbitrary points can be selected and they can be marked with a filled circle. Then their 3D positions (X, Y, Z) has to be written vertically inside a rectangular box, in $[X, Y, Z]^T$ format. They can be printed or handwritten and the rectangular does not have to perfect, the algorithm is able to detect it even it is distorted.

Figure 4.1: Test environment. The car is a representation of the object. The 3 circles around are the position of the 3 known points. Their corresponding 3D coordinates are written inside the rectangles.

4.1.2 Back-Projection Algorithm

To obtain the 3D coordinates of the 2D tracking point of the object back-projection, three points around the object are needed whose 2D and 3D locations are known. Those points should be around the object, on a plane platform and should be marked with a filled circle. Their corresponding 3D coordinates can be written in print form or by hand inside a rectangular box, where the coordinates should be written vertically in $[X, Y, Z]^T$ format. The marks and writings are color invariant.

Now, the three points - p, q, r around the object form a triangle Δpqr and if the object is represented with a single point c then as in fig:4.2 this point is inside the triangle. Then the point c can be represented by the three vertices of the triangle by linear interpolation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + t_1 \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} + t_2 \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \end{pmatrix} \tag{4.1}$$

or,

$$\begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix} = t_1 \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} + t_2 \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \end{pmatrix} \tag{4.2}$$

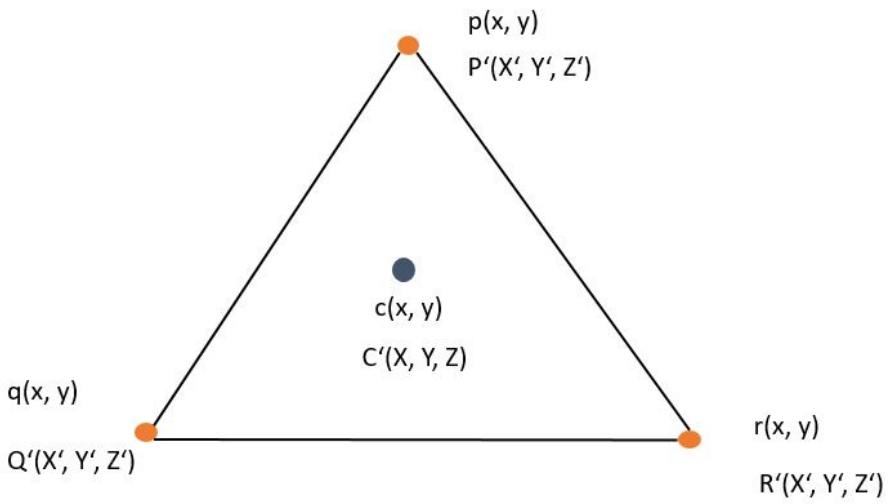


Figure 4.2: 3D reconstruction

or,

$$\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix} \tag{4.3}$$

From here we can get the translation coefficients t_1 and t_2,

$$t_1 = \frac{xy_3 - xy_1 - x_1y_3 + x_1y - x_3y + x_3y_1}{x_2y_3 - x_2y_1 - x_1y_3 - x_3y_2 + x_3y_1 + x_1y_2} \tag{4.4}$$

and

$$t_2 = \frac{x_2y - x_2y_1 - x_1y - xy_2 + xy_1 + x_1y_2}{x_2y_3 - x_2y_1 - x_1y_3 - x_3y_2 + x_3y_1 + x_1y_2} \tag{4.5}$$

Now, similar to the equation (4.1) we can also write for the 3D coordinates, using the value of t_1 and t_2 from (4.4) and (4.5) respectively:

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} X_1' \\ Y_1' \\ Z_1' \end{pmatrix} + t_1 \begin{pmatrix} X_2' - X_1' \\ Y_2' - Y_1' \\ Z_2' - Z_1' \end{pmatrix} + t_2 \begin{pmatrix} X_3' - X_1' \\ Y_3' - Y_1' \\ Z_3' - Z_1' \end{pmatrix} \tag{4.6}$$

So, we can write:

$$X' = X_1' + t_1(x,y)(X_2' - X_1') + t_2(x,y)(X_3' - X_1') \tag{4.7}$$
$$Y' = Y_1' + t_1(x,y)(Y_2' - Y_1') + t_2(x,y)(Y_3' - Y_1') \tag{4.8}$$
$$Z' = Z_1' + t_1(x,y)(Z_2' - Z_1') + t_2(x,y)(Z_3' - Z_1') \tag{4.9}$$

Therefore, given the input image, if the 2D and 3D coordinates of the circular marks and the 2D coordinates of the point inside the triangle can be detected, then using the above equation the corresponding 3D coordinates of the tracking point can be calculated.

4.2 Realization

This section discusses in detail how the algorithm has been implemented. It has been realized in two modes: manual and automatic. We will show how the program takes collects data in both mode and provide the output.

Manual implementation

The manual mode is mainly for test purpose, there is no object detection involved here. The inputs have to be provided manually by the user. Here, the user clicks on the screen provided by the video camera, to provide the three reference points and then the user chooses a point of interest or the tracking point, which is desired to be back-projected to 3D. The user also has to provide the corresponding 3D coordinates of those three points by writing. These data go through the back-projection algorithm and the corresponding 3D projected point of the tracking point is delivered.

Automatic implementation

In the automatic operation, all the required information is collected by the system automatically from the image-frame in real-time. The following algorithm (4.3) describe the high level functionality. The camera feed goes through the object detection module, which detected the object and finds a tracking point on it. The frames also go through the coordinate detection module, which detects the hand-written 3D coordinates and 2D position of the marks. In the end, all of this collected information is sent to the back-projection algorithm.

4.2.1 Object Detection

In 2.4 we have discussed Mask R-CNN instance segmentation model, which has been used here for Object detection. Mask R-CNN has been trained on COCO [30] data set, which has 91 object classes, consisting of many common objects. The reason we have chosen Mask R-CNN model over others is that till now, it is the most accurate object detection model. However, real-time object detection with Mask R-CNN is not possible in real-time using CPU, GPU is required. In that case, other object detection model can also be chosen. For this purpose, SSD or YOLO can be considered. YOLO has a remarkably high frame rate.

Besides, since it is an instance segmentation model and it provides a binary object mask as the output that makes finding the tracking point on the object easier. In our implementation, we have used the Mask R-CNN implementation by Abdulla [72].

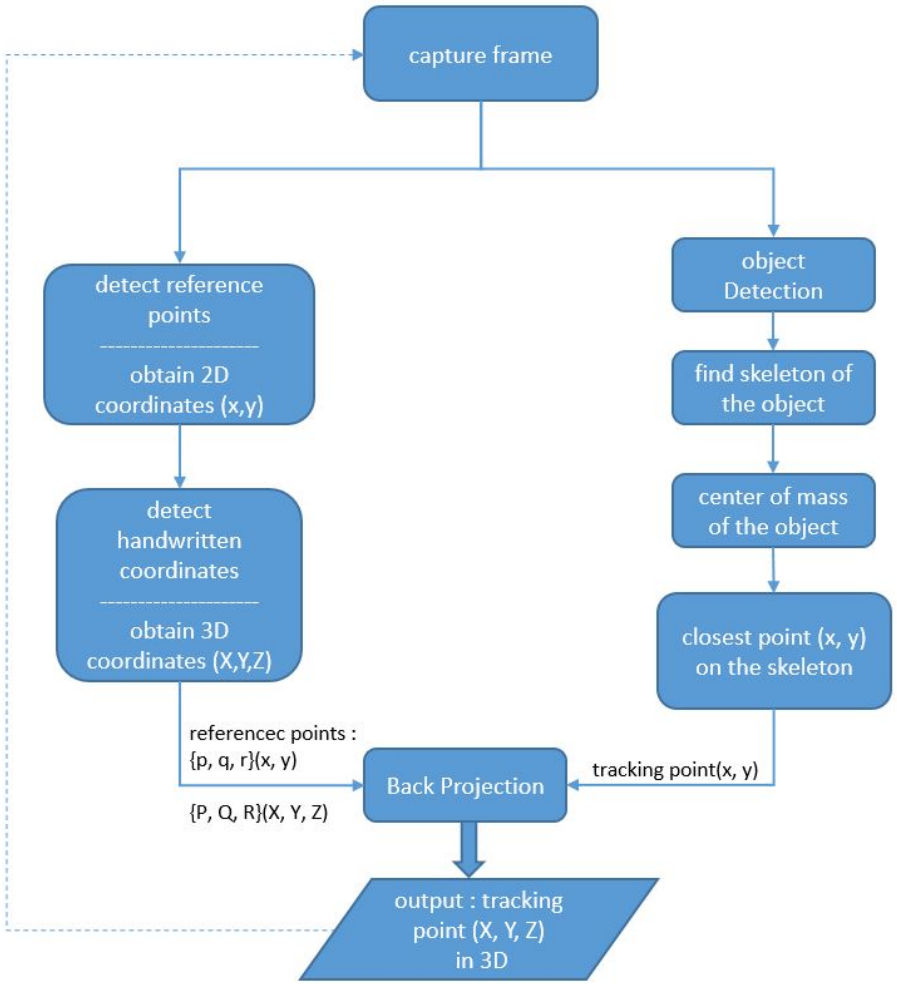


Figure 4.3: High level description of the proposed algorithm

Skeletonization

Once the object is detected, the next step is to extract a feature or a tracking point to track the object, since in the proposed back-projection algorithm the object is represented by a single point. One straight forward idea might be to choose the center of mass but if the object is of irregular shape or if it is a hollow object then the center of mass may not be on the body of the object. That is why it does not fulfill our purpose. Therefore, we plan to extract the skeleton of the object and then choose a point on the skeleton which is close to the centroid of the object mask.

The skeleton of the object is obtained through Zhang-Suen thining [73] algorithm. It is a fast parallel thinning algorithm that iteratively thins or removes the outer most pixels of the contour continuously until there is a one-pixel width skeleton is left. This procedure makes sure that the endpoints are preserved and the connectivity of the skeleton is maintained.

Here in fig 4.6 it shows the result of object detection by Mask R-CNN algorithm

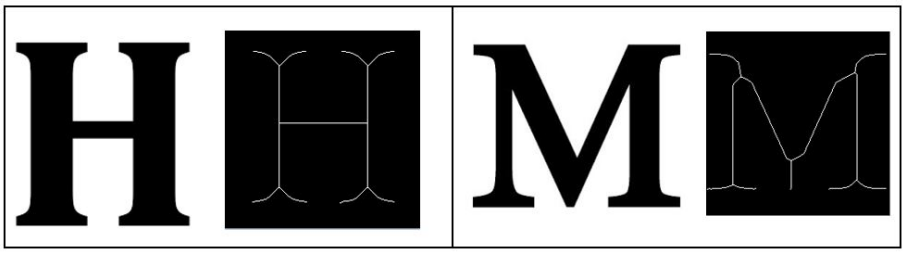


Figure 4.4: Examples of Zhang-Suen thining algorithm

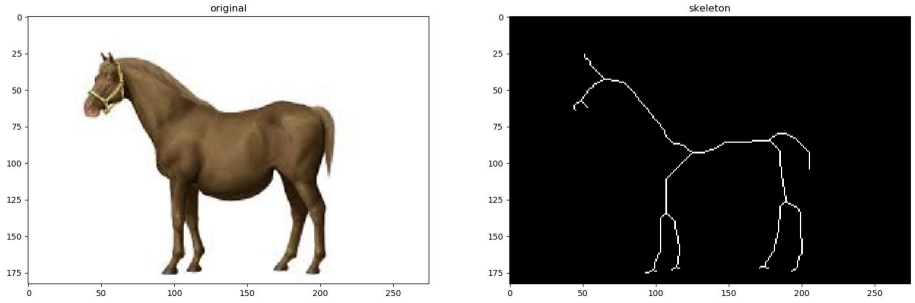


Figure 4.5: Examples of Zhang-Suen thinning algorithm

and then the skeleton has been created by the thinning algorithm of Zhang-Suen. Afterward, the center of mass has been calculated from the object mask and then its closest point is the point which is the desired point for tracking.

4.2.2 Back Projection

Now that the tracking point of the object or the point $c(x, y)$ in fig 4.2 is available, the back-projection algorithm requires the other three points of the triangle. The 2D coordinates can be obtained from the pixel location of the circles in the image. For 3D coordinates, after extracting the handwritten numbers from the boxes, each digit goes through a CNN classifier.

4.2.3 Circle detection

Each of the frames goes through the circle detection model, which followed by some preprocessing passes through hough circle transformation. To extract the right ones and avoid false detections, only those circles are selected which are close to the top-left edges of the box and the ratio of their area is also considered.

Coordinate detection : Preprocessing

Each 3D point is in a box close to the corresponding circle. Therefore, after blurring through edge detection, four corners of the boxes are detected. To provide some tolerance, so that the boxes do not have to be perfect, a polygon approximation

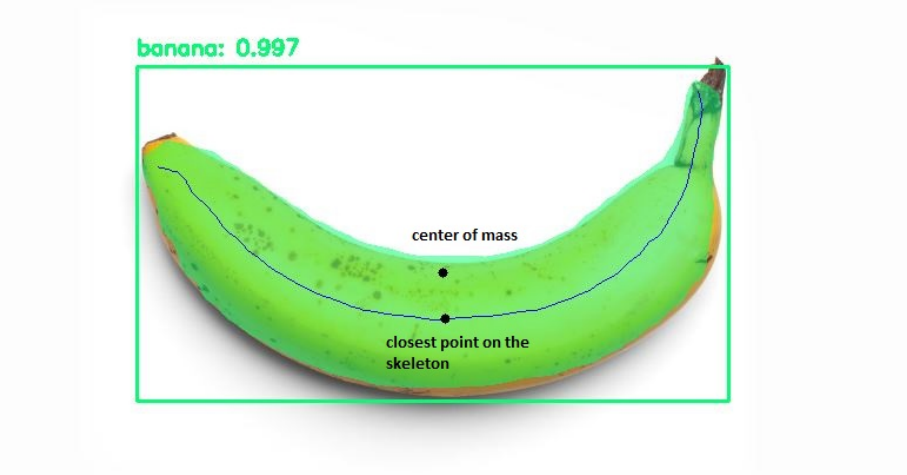


Figure 4.6: Object and object mask detection by Mask R-CNN. Skeleton from object mask. Tracking point based on center of mass and object skeleton.

function based on the Douglas-Peucker algorithm [74] has been used. After that, we have used perspective projection to warp the image and to bring them to the orthogonal bird's eye view. Now, even the camera or the numbers are moved and rotated almost about 45°, still, the boxes are detected correctly. The content of the boxes is cropped, thus they are ready now for number detection. Fig. 4.7 is showing the steps.

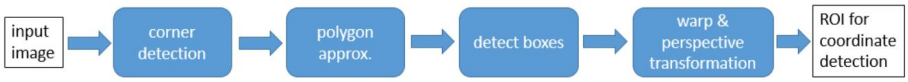


Figure 4.7: Algorithm for extracting the ROI for each coordinate detection.

Coordinate detection : CNN classifier

Since, it has been ensured at this point, that the content of the boxes, which are individual images, contain only digits. After preprocessing, edge detection, noise filtering according to area and size, the digits are extracted.

When the digits have been detected and cropped the digits individually from the boxes, they had to be preprocessed for the prediction according to the MNIST dataset. In MNIST dataset all the images have 28x28 pixel sizes, where the digit itself is of size 20x20 and shifted in the center. The digits have to be in white background and white foreground. After preparing the digits like this, they are passed through the classifier network. Fig. 4.8 is showing the algorithm and 4.9 shows an intermediate step.

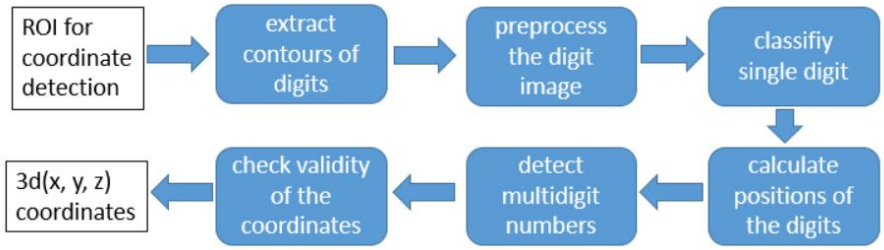


Figure 4.8: Steps for detecting and classifiying single digits and then detecting multi-digit single numbers, followed by group of coordinate detection.

Network Design

The network has been designed using state of the art features. Some of them will be shortly discussed here. The network consists of five convolution layers - the number of filters increases from 32 to 128 in the first three layers, then the last two layers have 160 filters each. There are two max-pooling layers for reducing overfitting and dimensionality.

The dropout layer has also been introduced, which is a regularization method. By dropping out or disabling neurons in the learning phase it reduces overfitting and forces the model to learn a different independent representation of the same data.

As optimizer, adam [75] has been chosen. With very little tuning it works very efficiently, with low memory consumption.

Arguably, the most important hyperparameter is the learning rate [11], which controls how quickly the model reacts to the problem. Smaller learning rate leads to slow convergence and large learning rate towards fast convergence with larger steps. This network uses an adaptive learning algorithm, which adjusts the rate based on the accuracy of the network. Alongside, data augmentation has also been used, where to reduce overfitting the data-set has expanded by randomly rotating, shifting and zooming the images.

The architecture of the CNN model of the classifier for digit detection was as follows: (fig: 4.10).

When the individual digits in a box are classified (0-9), their positions are calculated and based on their positions multi-digit single-numbers are detected and then they are labeled as coordinates.

4.2.4 GUI

A GUI (Graphical User Interface) has been developed for the convenience of the user. Tkinter library of python has been used to design the GUI. It has two modes of operations: manual and auto. Figure: 4.11 shows the GUI screen and figure 4.12 shows how the program accesses input and processes it, in both mode of operation.

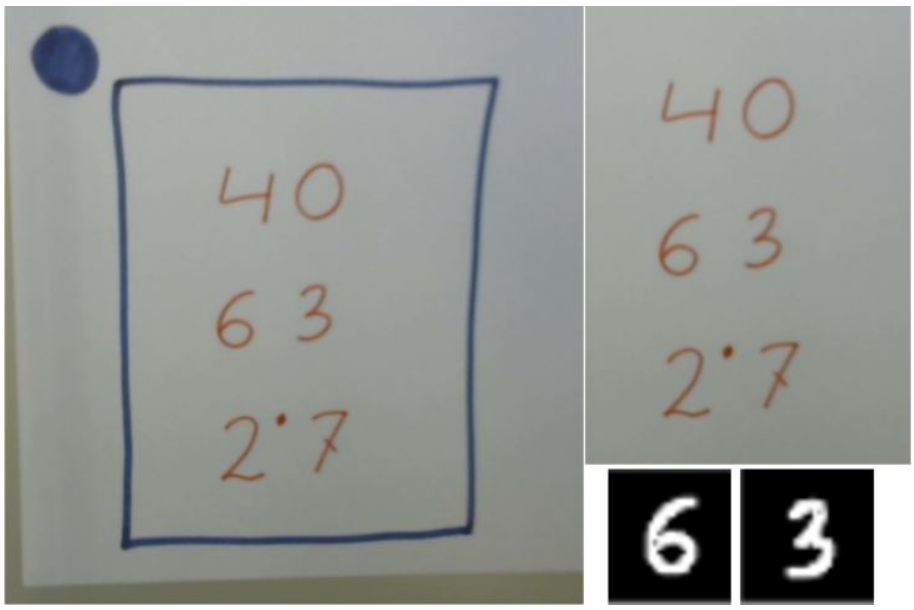


Figure 4.9: Preprocessing the digits for classification. At first, the four corners of the box is detected, then this ROI goes through warping and perspective transformation. After that, this processed ROI goes through the multi-digit number detection model, where the digits are classified individually, followed by number detection, based on their position.

Manual operation

In the manual mode, the user has to select three reference points from the scene by clicking on the screen. If the user moves the cursor, the real-time pixel position of the cursor can be seen on the screen and in the display label of the GUI as well. When the point will be chosen by clicking, its coordinates will stay on the screen. The corresponding 3D coordinates of the reference points need to be provided in the entry box by the user as well.

After choosing the three reference points, a triangle consisting of these three points will be shown. Now, the user has to select the fourth point or the tracking point inside the triangle, then the algorithm will calculate its equivalent three-dimensional coordinates and this will be displayed in the label. The user can test different inputs inside that triangle, meaning different input for the selected reference points. If the user wants to provide a new set of reference points it can be done so, by clicking **Manual** button again. Figure 4.13 shows the operation in action and figure 4.14 shows the details logic behind the manual mode operation.

Automatic operation

In the automatic mode, all the detection will be done automatically. The object will be detected with the help of the object detection model. From there, the skeleton and the tracking point will be calculated, which can be seen in the GUI screen and

display, respectively.

The reference points for back-projection are marked around the object. They are detected and their 2D coordinates are shown in real-time in the display. The 3D locations of the reference points are written inside rectangular boxes. These handwritten numbers are detected, their positions are checked against the reference points and hence displayed in the GUI labels as well.

With these data, the back-projection algorithm calculates the corresponding 3D coordinates of the tracking point and this can be seen in the label.

Now, if the user thinks the back-projection is done and now wants only to track the object, then the user can press the "only tracking" button. Then coordinate detection will stop and only the real-time 2D and 3D position of the object will be updated.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 20, 20, 64)        51264
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_8 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
dropout_12 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 6, 6, 160)         184480
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 3, 3, 160)         0
_________________________________________________________________
dropout_13 (Dropout)         (None, 3, 3, 160)         0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 1, 1, 160)         230560
_________________________________________________________________
batch_normalization_9 (Batch (None, 1, 1, 160)         640
_________________________________________________________________
dropout_14 (Dropout)         (None, 1, 1, 160)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 160)               0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               41216
_________________________________________________________________
dense_10 (Dense)             (None, 128)               32896
_________________________________________________________________
dense_11 (Dense)             (None, 10)                1290
=================================================================
Total params: 617,290
Trainable params: 616,842
Non-trainable params: 448
_________________________________________________________________
None
```

Figure 4.10: CNN architecture of the MNIST based keras classifier for digit detection.

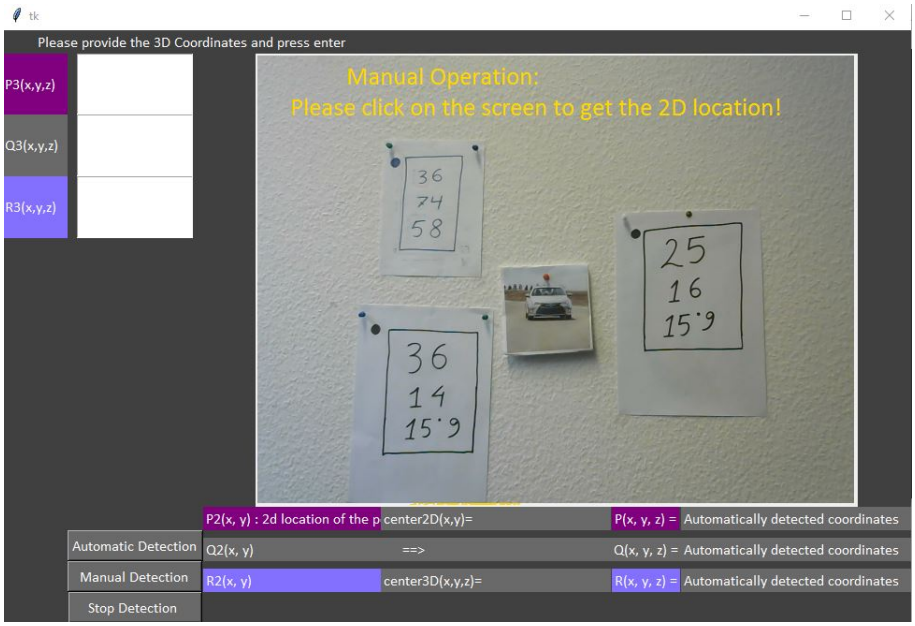


Figure 4.11: GUI of the framework

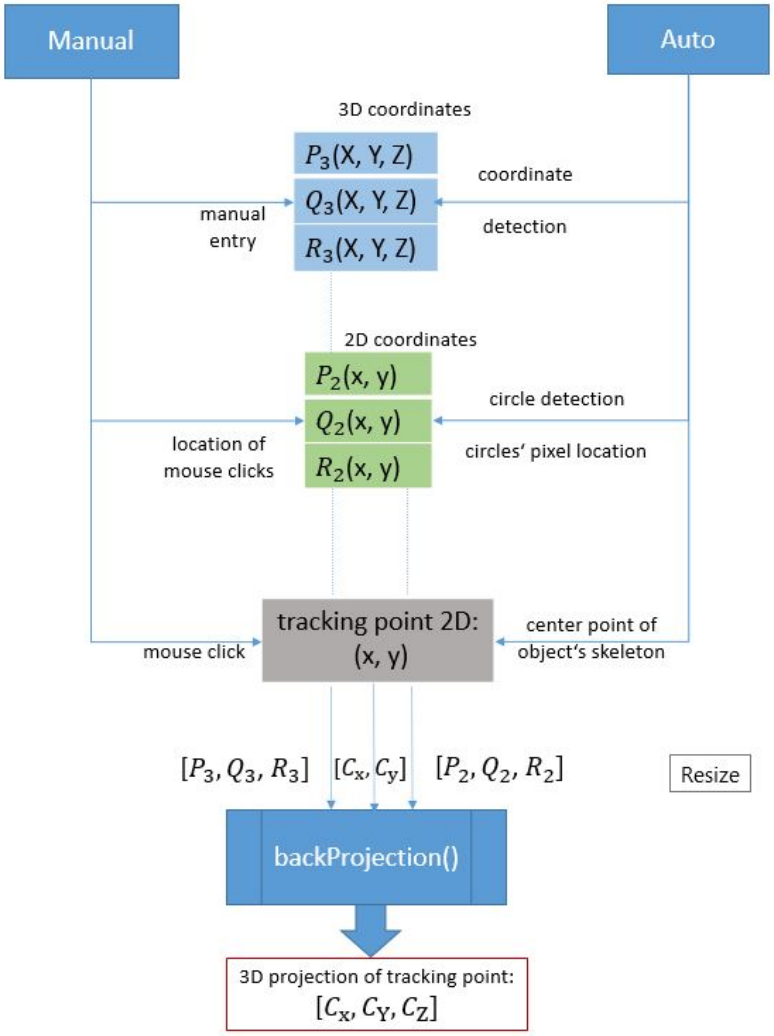


Figure 4.12: This diagram describes how both modes receive the input.
For example: in the manual mode the 3D coordinates are received through manual entry by the user, wherein the automatic mode, coordinate detection module provides these inputs by automatically detecting the handwritten 3D coordinates.
During manual operation, 2D coordinates of the reference points and also the 2D tracking point are obtained by the mouse clicks by the user on the screen. In auto operation, circle detection module detects the position of the reference points and the 2D tracking point comes from the object detection module.
At the end in both modes of operation, all the inputs are sent to the back-projection module which calculates the 3D projection point of the tracking point.

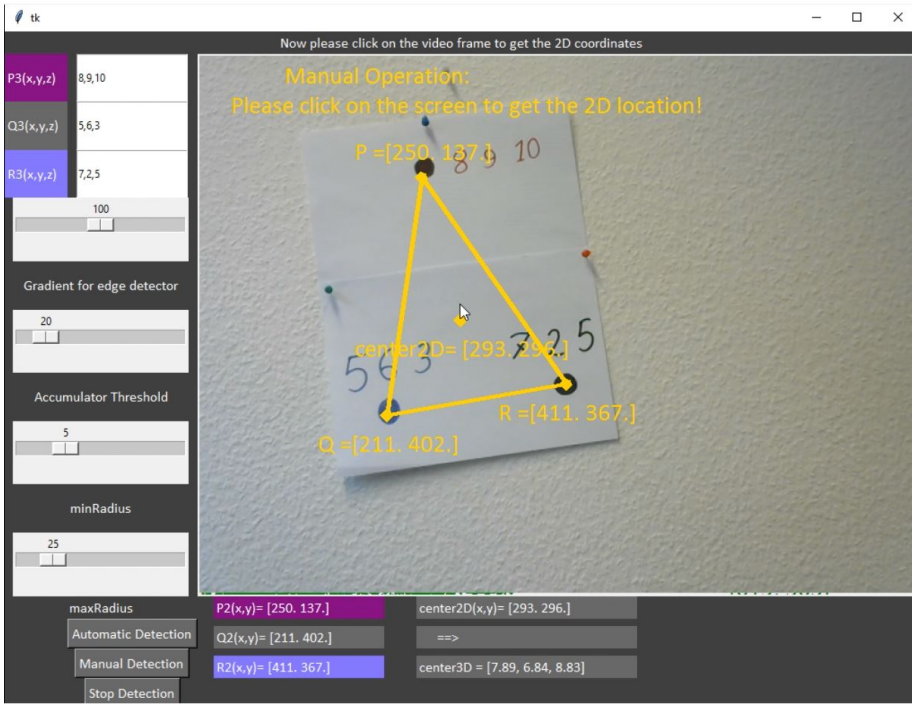


Figure 4.13: GUI operation : Manual Mode

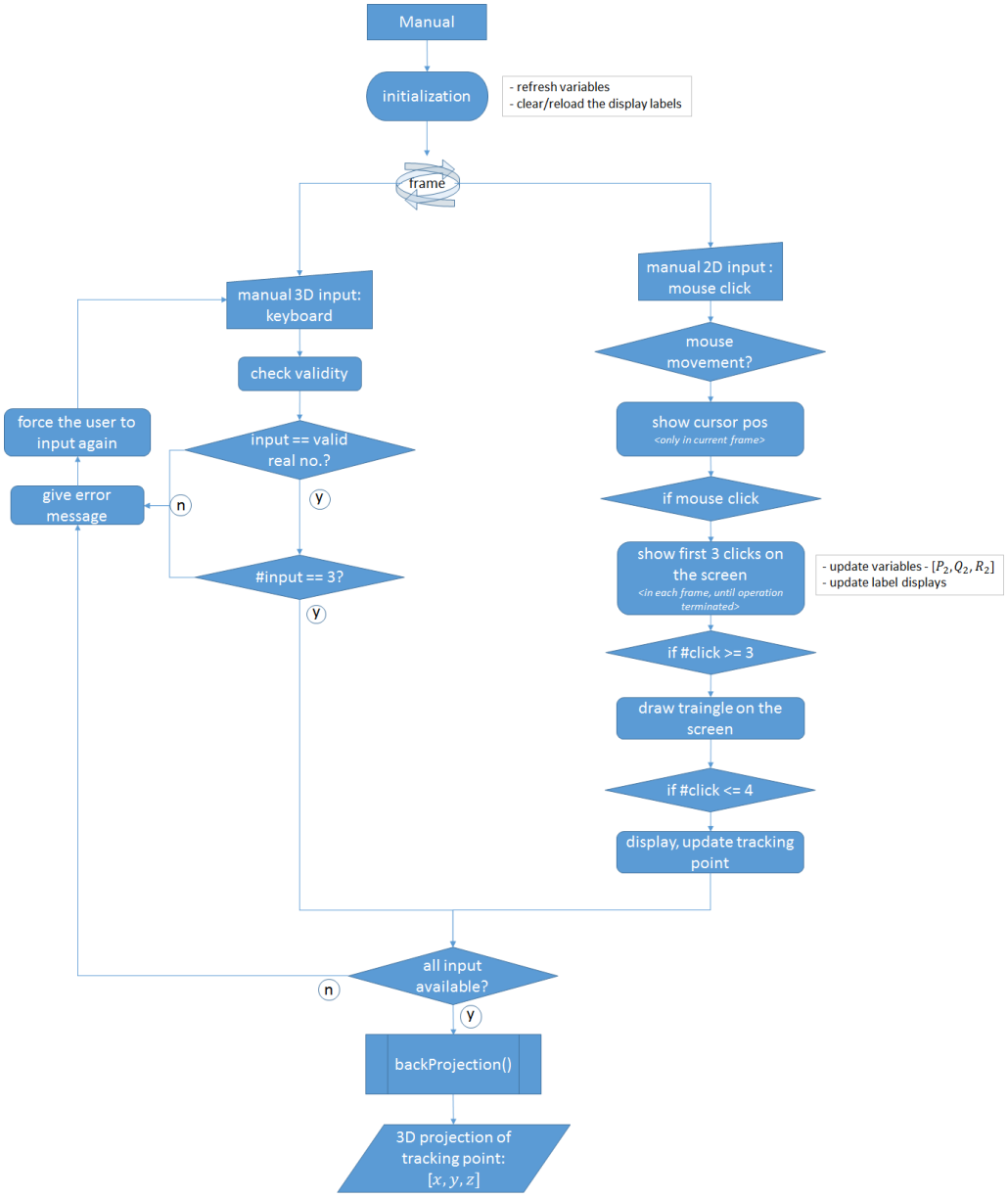


Figure 4.14: GUI operation : Algorithm for handling data during the manual mode.

5 Evaluation

This chapter analyzes the results of the algorithm. The limitations have been discussed and possible future improvement has been proposed.

The evaluation can be divided into two separate parts. Evaluating the object tracking model and evaluating the handwritten number detection model.

The objective of this thesis is to detect the 3D location of an object from a single 2D image. The object is detected using the object detection model and both the 2D and 3D position of three reference points around the object has to be detected. The 2D coordinates are detected by circle detection and the 3D coordinates are detected using the digit detection model.

For digit detection, a neural have been trained on MNIST dataset. It has achieved 99.77% accuracy on MNIST validation dataset. With three convolutional layers, the accuracy was 99.2%. The accuracy increased after increasing the number of convolutional layers up to 5. After 5 layers the accuracy goes down. We have also used data-augmentation which has also played a role in avoiding overfitting.

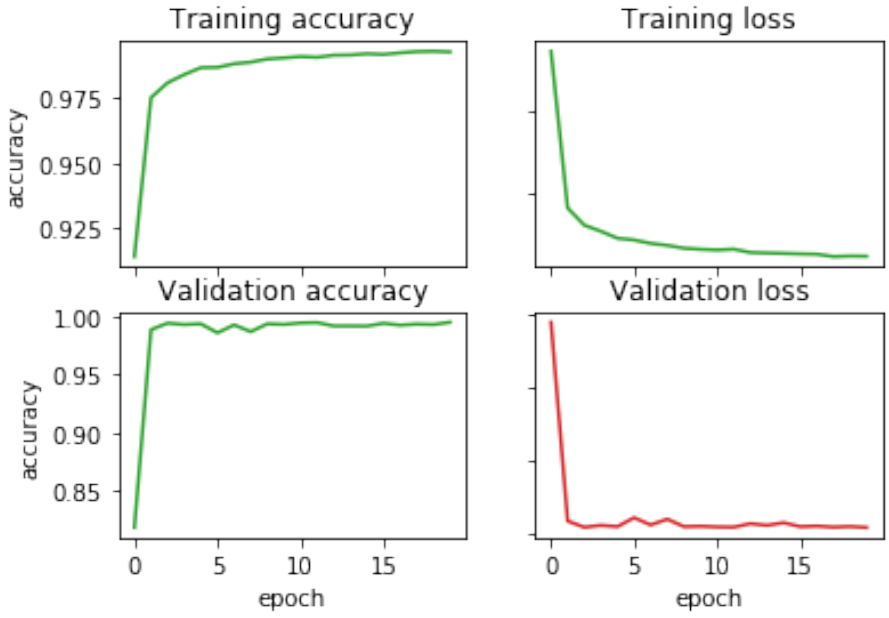


Figure 5.1: Performance of CNN model trained on MNIST dataset. Accuracy reached 99.77% in validation dataset.

The multi-digit number detection system has been developed based on this single digit detection model, where the numbers are detected based on their positions.

Figure 5.2 is showing a snapshot from real-time multi-digit number and coordinate detection.

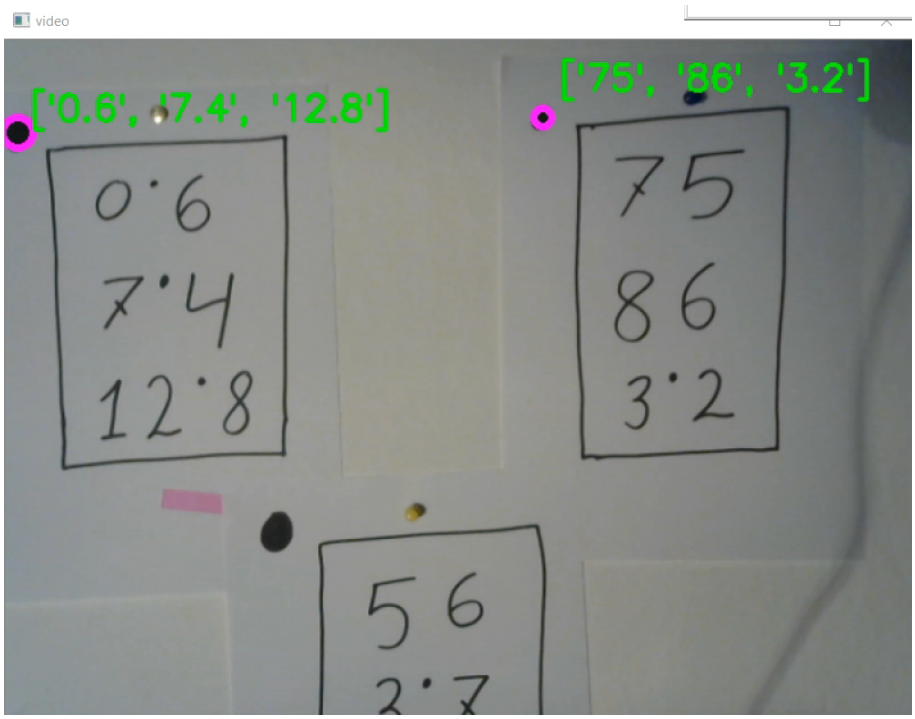


Figure 5.2: Snapshot from real-time multi-digit number and coordinate detection. Only the numbers which are inside the box are considered for coordinate detection.

Currently, the system works well when the camera is 38.2 inches or 3.2 feet (approx.) away from the object plane with camera rotation angle 45°in both horizontal and vertical directions. At this distance, the ideal box size is around 20cm, where the length of the digits should be approximately 3cm. The detection of box, circle and writings are color invariant, they can be written and drawn with any color.

The number recognition system has been designed in such a way that the 3D (x,y,z) coordinates of a point need to be vertically inside a rectangular box. Using this setting gives some advantages.

One advantage is the robustness to distortion. After detecting the ROI through the four corners of the box, the ROI is cropped, then warp and perspective transformation is performed and the ROI is brought back to orthogonal or bird's eye view. Then the content of the box goes through number detection. Therefore, due to the transformation, the box does not have to be perfectly square and also needs not to be in an orthogonal position with respect to the camera. Results 5.3 and 5.4 show that the detection works even if the camera or the boxes are rotated up to 45°, both horizontally and vertically.

Another advantage is the separation of the points from one another. In writing coordinates of a point are separated by commas (,) but MNIST dataset contains

only numerical digits, no symbols are included. Therefore, accurately detecting a handwritten comma would be challenging. On the other hand, keeping the coordinates of a point inside a box makes it easy to separate the point from another and then the coordinates are written vertically, so by calculating their vertical position, the x, y & z can be extracted.

Besides, this box based system also helps to avoid noise and unwanted detection. Since it only considers the numbers inside the boxes, so even if there are random numbers written outside, the algorithm will not consider them for detection.

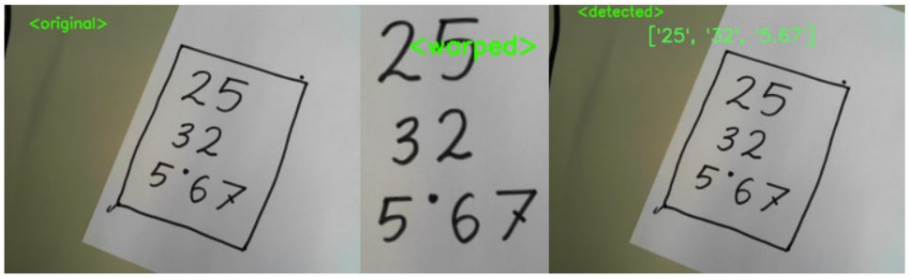


Figure 5.3: Coordinate detection, when the camera or numbers are rotated.

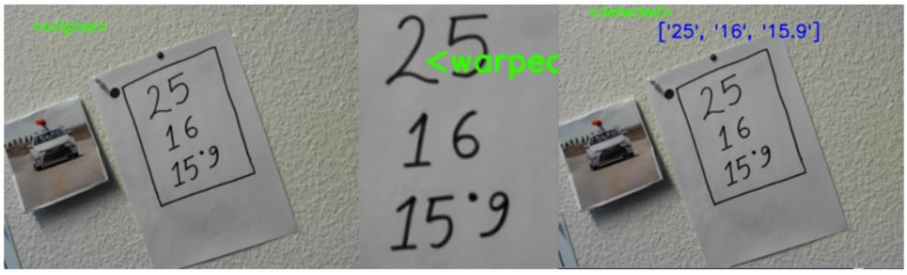


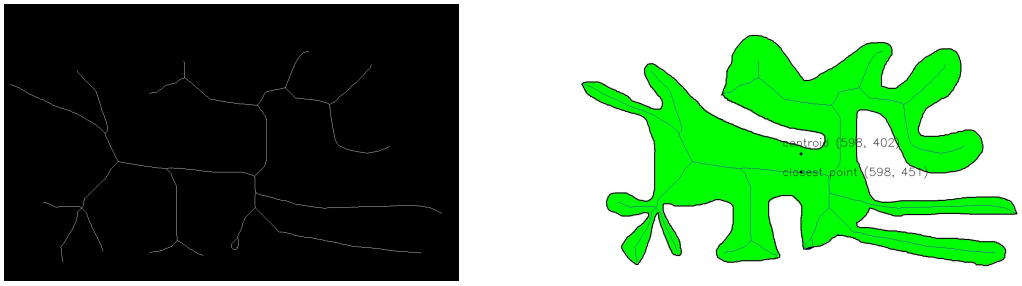
Figure 5.4: Coordinate detection, when the camera or numbers are rotated.

After detecting the object with Mask R-CNN or SSD, which are the state of the art model, the object is located in 3D with a relative center point or tracking point. The tracking point is obtained from the mask of the object, which is the output of Mask-RCNN. It can also be found from the mask of the contour of the object, since the mask output produced from Mask R-CNN architecture often does not precisely represent the shape of the object, especially in case of hollow objects.

To localize the tracking point, the skeletonization method has been used but we have also considered other methods. One of them was, the pole of inaccessibility [76] algorithm, which is also used in real-time mapping technology for localization purposes. In simple words, the pole of inaccessibility is the region in the area of interest, where the biggest circle can be inscribed. If optimized, this method is fast and suitable for real-time usage but the problem is, with respect to the camera, the biggest region is not always the same. Due to the perspective projection, when the object moves or the camera is in an angular position, the shape of the object will

be deformed, which means the region of the object which is closer to the camera becomes bigger. Thus, the tracking point based on this pole will also fluctuate.

Therefore, we have used the skeletonization method, which produces an accurate skeleton, while maintaining connectivity, see 4.4, 4.5. From the skeleton, the algorithm chooses the center point see 4.6 and it is serving the purpose well. This idea is more robust than the pole of inaccessibility algorithm against deformation. It works fine on objects of irregular and complex shape, fig.: (5.5).



(a) Skeleton of an object of highly irregular shape

(b) Tracking point or the closest point on the skeleton from the center of mass.

Figure 5.5: Generating skeleton by parallel thinking process and then computing the center of the skeleton.

However, change in the scale of the object will cause scale change in the skeleton as well, which will create instability in the tracking point. A possible good solution to this problem can be obtaining the skeleton from Mean Curvature Flow. Tagliasacchi [77] et al. have developed an algorithm for generating the skeleton of a 3D object. This also ensures a medially centered and connected curve skeletons. Mean curvature flow is equivalent to heat flow. By using inward surface evolution the skeleton can be achieved.

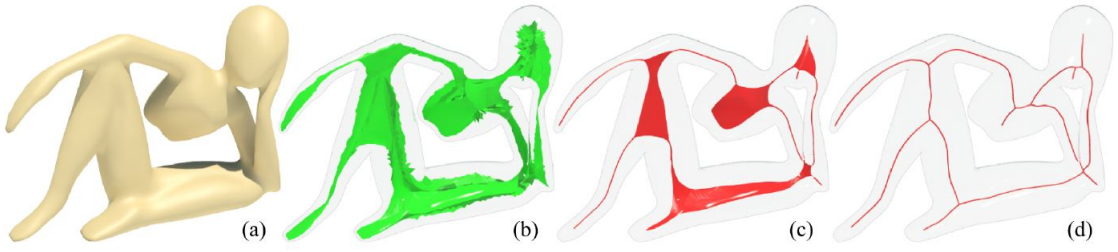


Figure 5.6: Obtaining skeleton of a 3D object using Mean Curvature Skeleton. [77]

The back-projection operation of the algorithm works as expected, but there is some difference between the 3D position of the projected point and the actual 3D position of the point on the object. First of all, all the reference points are on the same plane and the algorithm is using linear interpolation, so it works only when the object is moving on the reference plane because it back projects the tracking point

on the object on the reference plane. That is why, since it does not have information about the depth and the thickness of the object, the point is not projected on the surface of the object, which is shown in figure (5.7). For this reason, the thicker the object is, the higher will be the error, which is the difference between the projected point and the actual point.

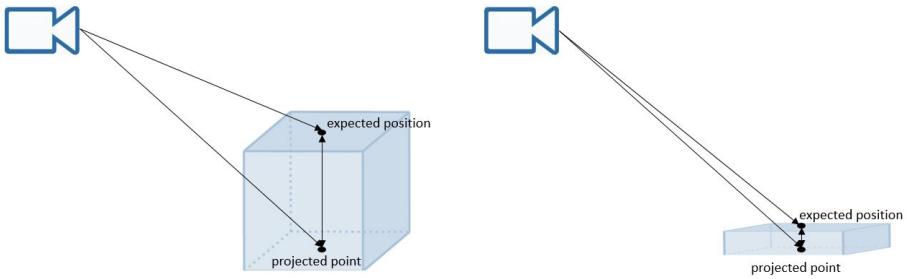


Figure 5.7: The algorithm back-projects the tracking point on the reference plane since it does not have depth information. Therefore, there is an error between the projected point and the actual point. The higher the thickness of the object is, the bigger the error.

This error will be even higher if the object leaves the reference plane.

6 Conclusion and Future Work

The aim of this thesis was to develop a real-time object detection and tracking algorithm in three dimensions, from a single image. It was also the target to develop an alternative of camera calibration because this is a single 2D image-based system and in a 2D image, depth information is lost, hence multiple images are needed and camera parameters have to be known as well. The proposed system does not need to know any camera parameters, it uses back projection, where three reference points around the object are used. The 2D positions of the reference points are collected from the image by detecting their marks, in this case, circle. The 3D coordinates are written either printed or in handwritten form. Therefore, one of the major challenges of this work was to detect handwritten numbers.

For that, a multi-digit number detection system has been developed. At its core, there is a CNN model, trained on the MNIST dataset. The coordinates of the points are written inside boxes, which gives the system the opportunity to be robust against rotation and distortion up to a point.

The object can be detected by Mask R-CNN, YOLO or SSD object detection model, which are one the most reliable models of the current time. Once the object is detected, it goes through a skeletonization algorithm, which uses a parallel thinning process and then a tracking point is selected on the skeleton. This point is back-projected to 3D. This skeletonization algorithm ensures that the tracking point is on the body of the object even if the object has a highly irregular shape.

To summarize the whole work it can be said that, the goal of the work has been achieved. An algorithm has been developed which can detect an object in 2D image and track in 3D using one single point. The 3D reconstruction through the back-projection algorithm is fast and reliable. However, if the camera is in an angular position then the algorithm is unable to recognize the depth of the object. In cases where the object is moving on a plane surface, this method can be useful for the detection and localization of the object.

Although it has limitations this work has created a foundation for a fast and simple method of detecting and tracking an object in 3D from a 2D image. Now there are scopes of improving it further and solving the challenges. For example, obtaining depth information, handling lens distortion and detecting the height of the object when the camera is in an angular position. The current method is unable to understand the orientation of the object and the tracking point may fluctuate in case of scaling since the point is dependent on the skeleton. Using Mean Curvature Flow for obtaining the skeleton and the center point of the object may make the system more reliable.

Bibliography

[1] H. Li and Y. Wang, "Object of interest tracking based on visual saliency and feature points matching," in *6th International Conference on Wireless, Mobile and Multi-Media (ICWMMN 2015)*, 2015, pp. 201–205.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[4] J. C. Plott, *Global History of Philosophy: The Period of scholasticism (part one)*, 1984.

[5] Wikipedia contributors, "Pinhole camera model — Wikipedia, the free encyclopedia," 2019, [Online; accessed 26-January-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Pinhole_camera_model&oldid=910238516

[6] N. Van Oosterwyck, "Real time human robot interactions and speed control of a robotic arm for collaborative operations," Ph.D. dissertation, 05 2018.

[7] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, 2000. [Online]. Available: https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/

[8] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 016–154, 1962. [Online]. Available: https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1962.sp006837

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[10] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *From Natural to Artificial Neural Computation*. Springer Berlin Heidelberg, 1995, pp. 195–201.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Omnipress, 2010, pp. 807–814. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104425

[13] Y. LeCun and Y. Bengio, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[14] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[16] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[18] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017. [Online]. Available: https://books.google.de/books?id=9Ul-tgEACAAJ

[19] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, cite arxiv:1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[22] J. Fu and Y. Rui, "Advances in deep learning approaches for image tagging," *APSIPA Transactions on Signal and Information Processing*, vol. 6, 10 2017.

[23] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 580–587. [Online]. Available: https://doi.org/10.1109/CVPR.2014.81

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[26] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] ——, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[28] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, "A review on deep learning techniques applied to semantic segmentation," *CoRR*, vol. abs/1704.06857, 2017. [Online]. Available: http://arxiv.org/abs/1704.06857

[29] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2980–2988.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. European Conference on Computer Vision, September 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/microsoft-coco-common-objects-in-context/

[31] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[32] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. IEEE Computer Society, 2015, pp. 1440–1448. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.169

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 91–99. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969250

[34] X. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *CoRR*, vol. abs/1906.06543, 2019. [Online]. Available: http://arxiv.org/abs/1906.06543

[35] M. Gadelha, S. Maji, and R. Wang, "3d shape induction from 2d views of multiple objects," in *2017 International Conference on 3D Vision (3DV)*, Oct 2017, pp. 402–411.

[36] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 628–644.

[37] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[38] K.-m. G. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette across time part i: Theory and algorithms," *International Journal of Computer Vision*, vol. 62, no. 3, pp. 221–247, May 2005. [Online]. Available: https://doi.org/10.1007/s11263-005-4881-5

[39] K. M. G. Cheung, S. Baker, and T. Kanade, "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, June 2003, pp. I–I.

[40] G. K. M. Cheung, S. Baker, and T. Kanade, "Visual hull alignment and refinement across time: a 3d reconstruction algorithm combining shape-from-silhouette with stereo," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, June 2003, pp. II–375.

[41] W. Phothong, T.-C. Wu, J.-Y. Lai, D. W. Wang, C.-Y. Liao, and J.-Y. Lee, "Fast and accurate triangular model generation for the shape-from-silhouette technique," *Computer-Aided Design and Applications*, vol. 14, no. 4, pp. 436–449, 2017. [Online]. Available: https://doi.org/10.1080/16864360.2016.1257186

[42] Y. Xiang, S. Nakamura, H. Tamari, S. Takano, and Y. Okada, "3d model generation of cattle by shape-from-silhouette method for ict agriculture," in *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, July 2016, pp. 611–616.

[43] M. Wilczkowiak, E. Boyer, and P. Sturm, "3d modelling using geometric constraints: A parallelepiped based approach," in *Computer Vision — ECCV 2002*, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 221–236.

[44] M. Wilczkowiak, P. Sturm, and E. Boyer, "Using geometric constraints through parallelepipeds for calibration and 3d modeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 194–207, Feb 2005.

[45] M. Wilczkowiak, E. Boyer, and P. Sturm, "Camera calibration and 3d reconstruction from single images using parallelepipeds," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, July 2001, pp. 142–148 vol.1.

[46] J. Deutscher, M. Isard, and J. MacCormick, "Automatic camera calibration from a single manhattan image," in *Computer Vision — ECCV 2002*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 175–188.

[47] S. Park, H. Lee, S. Lee, and H. S. Yang, "Line-based single view 3d reconstruction in manhattan world for augmented reality," in *Proceedings of the 14th ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, ser. VRCAI '15. New York, NY, USA: ACM, 2015, pp. 89–92. [Online]. Available: http://doi.acm.org/10.1145/2817675.2817689

[48] S. Ranade and S. Ramalingam, "Novel single view constraints for manhattan 3d line reconstruction," in *2018 International Conference on 3D Vision (3DV)*, Sep. 2018, pp. 625–633.

[49] E. Guillou, D. Meneveaux, E. Maisel, and K. Bouatouch, "Using vanishing points for camera calibration and coarse 3d reconstruction from a single image," *The Visual Computer*, vol. 16, no. 7, pp. 396–410, Nov 2000. [Online]. Available: https://doi.org/10.1007/PL00013394

[50] Q. Wu, T.-C. Shao, and T. Chen, "Robust self-calibration from single image using ransac," in *Advances in Visual Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 230–237.

[51] I. Miyagawa, H. Arai, and H. Koike, "Simple camera calibration from a single image using five points on two orthogonal 1-d objects," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1528–1538, June 2010.

[52] W. Zhang, S. Lin, F. H. Bijarbooneh, H. F. Cheng, and P. Hui, "Cloudar: A cloud-based framework for mobile augmented reality," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, ser. Thematic Workshops '17. New York, NY, USA: ACM, 2017, pp. 194–200. [Online]. Available: http://doi.acm.org/10.1145/3126686.3126739

[53] L. Fan, Z. Wang, B. Cail, C. Tao, Z. Zhang, Y. Wang, S. Li, F. Huang, S. Fu, and F. Zhang, "A survey on multiple object tracking algorithm," in *2016 IEEE International Conference on Information and Automation (ICIA)*, Aug 2016, pp. 1855–1862.

[54] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355

[55] W. Kloihofer and M. Kampel, "Interest point based tracking," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 3549–3552.

[56] P. Tissainayagam and D. Suter, "Object tracking in image sequences using point features," *Pattern Recognition*, vol. 38, no. 1, pp. 105 – 113, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320304002407

[57] G. Zhu, Q. Zeng, and C. Wang, "Efficient edge-based object tracking," *Pattern Recognition*, vol. 39, no. 11, pp. 2223 – 2226, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320306001737

[58] C. Shen, J. Kim, and H. Wang, "Generalized kernel-based visual tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp. 119–130, Jan 2010.

[59] Z. Fan, Y. Wu, and M. Yang, "Multiple collaborative kernel tracking," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, June 2005, pp. 502–509 vol. 2.

[60] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, Feb 2001.

[61] A. Yilmaz, X. Li, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1531–1536, Nov 2004.

[62] W. Hu, X. Zhou, W. Li, W. Luo, X. Zhang, and S. Maybank, "Active contour-based visual tracking by integrating colors, shapes, and motions," *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1778–1792, May 2013.

[63] M. Yokoyama and T. Poggio, "A contour-based moving object detection and tracking," in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Oct 2005, pp. 271–276.

[64] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[65] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," 2013. [Online]. Available: http://ufldl.stanford.edu/housenumbers

[66] U. R. Babu, Y. Venkateswarlu, and A. K. Chintha, "Handwritten digit recognition using k-nearest neighbour classifier," in *2014 World Congress on Computing and Communication Technologies*, Feb 2014, pp. 60–65.

[67] E. Tuba and N. Bacanin, "An algorithm for handwritten digit recognition using projection histograms and svm classifier," in *2015 23rd Telecommunications Forum Telfor (TELFOR)*, Nov 2015, pp. 464–467.

[68] X. Gao, B. Guan, and L. Yu, "Handwritten digit recognition based on support vector machine," in *First International Conference on Information Sciences, Machinery, Materials and Energy*. Atlantis Press, 2015/07. [Online]. Available: https://doi.org/10.2991/icismme-15.2015.198

[69] A. Boukharouba and A. Bennia, "Novel feature extraction technique for the recognition of handwritten digits," *Applied Computing and Informatics*, vol. 13, no. 1, pp. 19 – 26, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S221083271500006X

[70] K. T. Islam, G. Mujtaba, R. G. Raj, and H. F. Nweke, "Handwritten digits recognition with artificial neural network," in *2017 International Conference on Engineering Technology and Technopreneurship (ICE2T)*, Sep. 2017, pp. 1–4.

[71] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[72] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[73] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984. [Online]. Available: http://doi.acm.org/10.1145/357994.358023

[74] S. . Wu and M. R. G. Marquez, "A non-self-intersection douglas-peucker algorithm," in *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, Oct 2003, pp. 60–66.

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[76] D. Garcia-Castellanos and U. Lombardo, "Poles of inaccessibility: A calculation algorithm for the remotest places on earth," *Scottish Geographical Journal*, vol. 123, no. 3, pp. 227–233, 2007. [Online]. Available: https://doi.org/10.1080/14702540801897809

[77] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang, "Mean curvature skeletons," *Comput. Graph. Forum*, vol. 31, no. 5, p. 1735–1744, Aug. 2012. [Online]. Available: https://doi.org/10.1111/j.1467-8659.2012.03178.x