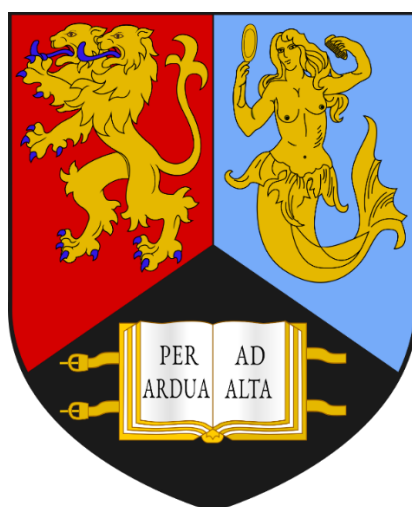


GAZED OBJECT - DEPTH ESTIMATION FOR MONOCULAR VIDEO SEQUENCES

GIRISH KALYAN JALASUTRAM

STUDENT ID: 2216842



SUPERVISOR: SANG-HOON YEO

A thesis submitted to University of Birmingham for the degree

MSc Artificial Intelligence and Machine Learning

School of Computer Science

University of Birmingham

Table of Contents	
Abstract	5
CHAPTER 1: Overview	6
1.1 Introduction	6
1.2 Gaze System	6
1.3 Motivation.....	7
1.4 Problem Statement	8
1.5 Objective	8
CHAPTER 2: Literature Review.....	9
2.1 Introduction	9
2.2 Overview of Machine learning	9
2.2.1 Importance of Machine Learning.....	9
2.2.2 Classification of ML Techniques.....	9
2.2 Review of Segmentation Technique.....	10
2.3 Review of Depth Estimation Technique	10
2.4 Gaze Eye Detection	10
2.5 Gaze Model	11
2.6 Object Detection Technique.....	11
2.7 Summary	12
Chapter 3: Methodology	13
3.1 Introduction	13
3.2 Proposed Method	13
3.2.1 Training Model.....	13
3.2.2 Multi-scale Estimation	14
3.3 Construction of Monodepth2.....	14
3.4 Dataset	14
3.5 The SSD model for Object Detection.....	14
3.6 Summary	15
Chapter 4: Result and Discussion	16
4.1 Introduction	16
4.2 Simulation Code	16
4.3 Constructed Model.....	18
4.4 Simulation Results	19
4.4.1 Object Detection Results	19
4.4.2 Result of Gaze-Depth Estimation Mode.....	23
4.5 Summary	24
Chapter 5: Conclusion and Future works	25
References	26
Appendix: GitLab Repository	29

List of Table and Figures	
Fig 1.1 – HCI based system	6
Fig 1.2 – Overview of Gaze System.	7
Fig 3.1 – Structure of the sparsity-invariant autoencoder.	14
Fig 3.2- Architecture of SSD Model	15
Fig 4.1(a)-Loading Monodepth model	17
Fig 4.1(b)- Depth map	18
Fig 4.1(c)- Object Detection Evaluation	19
Fig 4.2 -Object Detection Evaluation	20
Fig 4.3 – Training loss with 100 epochs	20
Fig 4.4 – Training accuracy with 100 epochs	21
Fig 4.5 – Testing loss with 100 epochs	22
Fig 4.6 – Testing accuracy with 100 epochs	23
Fig 4.7 – Training on Google Collab	23
Fig 4.8 – Gazed object depth estimation	24
Fig 4.9- Elapsed time graph	25
Table 1. Comparison between baseline models for depth estimation	24
Table 2 – Comparison between state of art models for depth estimation	24

Acknowledgement

I sincerely thank my Supervisor Prof Sang-Hoon Yeo for assisting and providing the opportunity to work on this project. The feedbacks provided by Prof Sang-Hoon Yeo were helpful to complete this project. And I would also like to thank my parents and friends for encouraging and providing mental support throughout the term. Without them, I wouldn't be able to complete this project.

Abstract

A depth-gaze analysis is one of the crucial requirements for self-driving cars since the model needs to mimic human navigation behavior without committing accidents. For that purpose, the software needs to learn the navigation on roads. The developed model utilizes existing reliable models for depth estimation and object detection and, later comparing these values concerning navigation speed helps in an understanding driving scenario in real time world.

In my model, my aim is to estimate the gazed object and calculate the distance between object and viewer, simply it's a Gaze depth estimation with object detection. I used KITTI dataset for training on modepth2 model for depth estimation part and SSD model for object detection part, the result that I got from these two step processes is efficient for gazed object depth estimation. I was able to efficiently estimate these values in seconds which are crucial for avoiding navigation errors. Monodepth model is able to estimate the distance with a minimum RMSE of 4.863 and SSD is able to predict the object with an accuracy of 96.3%

CHAPTER 1: Overview

1.1 Introduction

One of the emerging technologies and the latest field of research that encompasses more beneficial and meaningful interactions between humans and computers is Human-Computer-Interaction (HCI). HCI is the systematic study of algorithmic processes, mainly concerned with the interactive computing systems for human use that describe and transform information in terms of theory, analysis, design, efficiency, implementation, and application [1-3]. The necessity and significance of HCI research arise due to the limitations of the traditional methods of input devices like keyboards, mice, scanners, etc. A range of techniques can be used for hands free operations in HCI. These techniques may use eye gaze, gesture, facial recognition, head movement, speech recognition, etc., and can be used as a tool enabling hands-free operation of the display for the user, unlike the traditional input devices. The traditional methods of input devices have specific domains and proximity, thus cannot be used for a certain population of users. Further, the traditional devices may be difficult to be used if the hands are busy, or due to different medical reasons including neurological or muscular problems. Different forms of HCI devices input devices are motion sensing with hand, head, face, and eye movement as shown in Fig. 1.1.

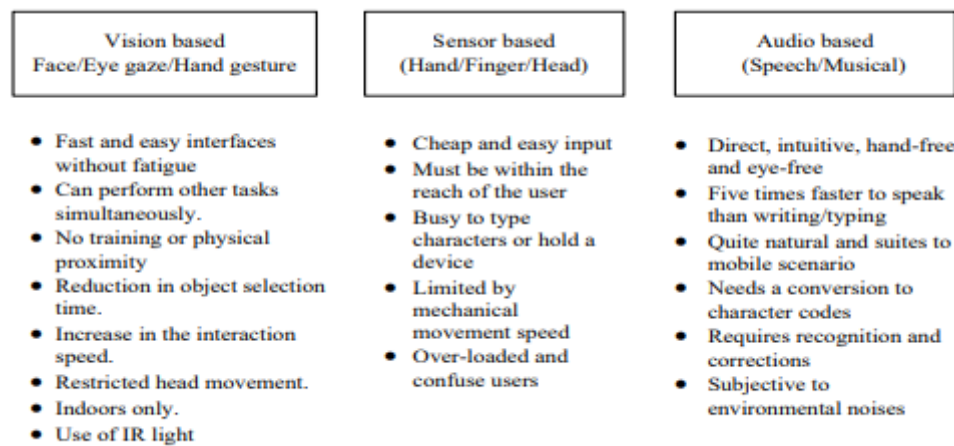


Fig 1.1: HCI based system

The audio-based device is controlled by the human voice. The spoken words are converted to text with the help of speech recognition devices. HCI and visualization form an important part of our daily work life. Computer visualization is a transformation of data into geometric shapes for better overview or insight into data high dimensions. On a basic level, HCI and visualization intersect in the usage of cognitive processes that represent data in geometric forms. Vision based HCI is a more instinctive way to manipulate data and accepts input just by seeing an object in the fields like hand gesture recognition, emotion recognition from face and eye gaze [4-6].

1.2 Gaze System

Robust eye detection and tracking are very significant in creating attentive user interfaces with the help of different eye gaze detection and estimation methods. In eye gaze detection, a system needs to read eye movements and then map them to a computer. The significance of eye gaze systems includes communication from a distance without any eye fatigue, training, coordination, or physical proximity with the computer. Eye gaze interaction is safer, resulting in a reduction in object selection time and an increase in the interaction speed [7-9]. Eye gaze refers to measuring the gaze direction with the help of Point of Gaze (PoG) or Point of Regard (PoR) or Region of interest (RoI) from the eye movements using static or dynamic images. The gaze direction is used to determine the user's intention or desire which helps in understanding, processing, and executing different commands from a distinct location. Moreover, gaze estimation requires the segmentation or extraction of local features like the eye outline, eye contours, edges of pupil or iris, eye corners, the centre of either iris or pupil, or corneal reflections or glint [10-12]. From the eye images as shown in Fig. 1.2., there may be multiple glints G_i formation due to multiple sources of incident light falling on the pupil region along with iris, sclera, etc. The relative position of the iris centre (C_i) with the centre of the eye (C_e) can be used for further gaze-based

processing to estimate the user's point of gaze. The process of eye gaze involves the user looking at the specific location on the screen and the eye image is captured using a single or multiple digital cameras or a webcam etc. in any light or infrared (IR) light sources. The user's gaze point on the screen is then estimated as shown in Fig. 1.2.

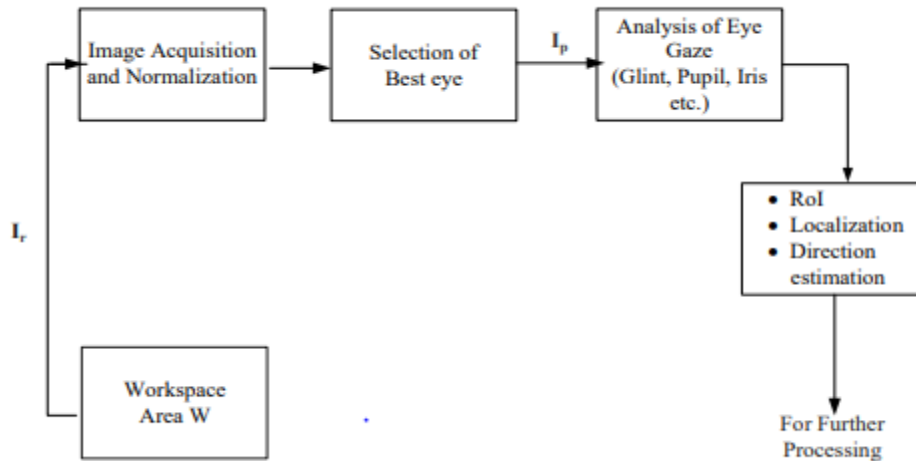


Figure 1.2: Overview of Gaze System

The captured raw image I_r is processed in the second phase from acquisition to estimation and is normalized for adjustment of colour tone, cropping, format conversion. Identification and selection of the best eye in the next phase are carried out based on different parameters.

Processed image I_p thus generated is then input to eye gaze model for the segmentation and extraction of RoI. RoI was used in the calculation of the gaze direction of eye gaze for further processing. The direction of gaze is normally detected by estimating the relative position of RoI by mapping reference points like glint or iris vector with the centre of the pupil or iris centre coordinates using any of the feature segmentation techniques [13-17]. Frequently used feature extraction techniques for the segmentation of the RoI from the eye image for gaze estimation are discussed in the next chapter. Another factor that plays a crucial role in eye gaze-based models is interactivity or response time. This factor is a critical measure for online real time systems. CPU interactivity, dwell and the profile time is important aspects to analyse interactivity. The efficiency of eye gaze-based models may improve and enhance the productivity of the gaze-based systems with low, minimum interactivity or response time along with profile time [18-21].

1.3 Motivation

Difficult calibration techniques, use of multiple cameras, costly eye trackers lead to more computational cost and complexity in the eye gaze systems. An environment with IR light, incandescent light bulbs, controlled visible light, head movements, and control parameters are some of the research issues that may affect the results of the eye gaze systems. Misinterpretation by the gaze interface can trigger unwanted actions and might result in conflicts [11,27,30,31,32]. Full facial images also lead to more time-consuming gaze systems. Accuracy of object or point in focus from the captured user's image is very crucial in deciding further course of action by the system. The short distance between the camera and the subject is also another research issue in the eye gaze-based models as the existing gaze detection systems work only over shorter distances up to a maximum of 70 cm leading to proximity constraint [33-36]. Full facial images also lead to more time-consuming gaze systems. Accuracy of object or point in focus from the captured user's image is very crucial in deciding further course of action by the system. In addition, factors like shape and size of the object, blur images, light from single or multiple sources, colour, orientation, the height of capturing devices, thresholds, response time, environmental set up etc. play an important role in the designing of effective eye gaze systems [37]. The selection of specific segmentation techniques also requires careful selection while designing these systems. The convenience of the user is another important research issue. The research focus is, therefore, to reduce the requirement of

expensive devices and explore the utilization of ubiquitous hardware and software at a relatively lower cost with fast interactivity time for gaze detection models.

1.4 Problem Statement

The depth-gaze analysis is estimated based on the sampled depth as per point of regard (PoR) based on the image space. The perfect calibration is required to withstand the accurate computation of gaze depth. However, conventional gaze systems are subjected to inaccuracies due to variation in angle, estimation, and fraction. Those parameters lead to false gaze estimation. It is necessary to develop an appropriate gaze depth estimation model for the accurate computation of objection. Also, in existing classification based on estimated gaze is not presented.

These research issues are highlighted by various researchers in the eye gaze tracking systems and have been discussed in the second chapter of the thesis. Investigation of the CPU time required by the gaze systems affects the efficiency and speed of the models and may, therefore, be analysed for building real time gaze applications. The least CPU time in the gaze-based models is essential for optimizing the efficiency and the response time of the gaze models [18-21,28].

Based on the above discussion, it has been observed that there is a need to further design optimized and improved eye gaze detection models having better accuracy and interactivity keeping the computational complexity minimum. Optimization is possible in terms of specific controlled parameters like distance, size, shape, orientation of the object, and capturing devices. If these parameters are studied, analysed, and further optimized, navigation behaviour is efficiently achieved.

1.5 Objective

This research aimed to develop an effective artificial intelligence system for gaze-depth estimation and classification mode monocular objects. The specific objective of this research is presented as follows:

1. To fine tune the monodepth2 model on KITTI Dataset
2. To optimise this model for more accurate and fast responses
3. To use this model for real time gaze-depth estimation

This research highly concentrated on monocular object detection video sequences. Therefore, I am going to optimise the monodepth2 model and finetune this model on DDAD Dataset so that it can be used for real time detection.

CHAPTER 2: Literature Review

2.1 Introduction

Different researchers have studied and proposed various methods and techniques of HCI like head pose estimation and eye gaze estimation and detection. In HCI, various active and passive approaches help to locate the exact position of the eye, head, face, etc. Various eye gaze-based models, algorithms, techniques of several researchers for understanding the different developments and innovations in the field of eye gaze estimation have been surveyed and observations have been made in this chapter.

2.2 Overview of Machine learning

Learning as a conventional cycle is tied in with obtaining new, or altering existing, practices, values, information, abilities, or inclinations. Behaviourism, Cognitivism, Constructivism, Experientialism, what's more, Social Learning characterizes the hypothesis of individual learning, for example how people learn. Machines depend on information in opposition to what falls into place without any issues for people: gaining as a matter of fact. At the exceptionally essential level AI (ML) is a class of man-made consciousness that empowers PCs to think and learn all alone. It is tied in with causing PCs to change their activities to work on the activities to achieve more precision, where exactness is estimated as far as the occasions times the picked activities results into right ones.

Certifiable problems are very complex, making them ideal for ML. For example, email spam filtering, friendly network extortion location, online stock trading, face and form discovery, clinical conclusion, traffic expectation, character recognition, and item proposal are all applications of AI. Online recommendation engines—like companion ideas on Facebook, "additional things to ponder" and "buy yourself a little bit" on Amazon, as well as Visa extortion location—are all examples of AI use.

2.2.1 Importance of Machine Learning

Certifiable problems have a high level of complexity, which makes them an excellent candidate for machine learning applications. To plan and program express calculations with superior yield, artificial intelligence (AI) can be applied to a variety of figuring spaces. Examples include email spam sifting, extortion location on friendly networks, online stock exchanging, face and shape discovery, clinical decision-making, traffic expectation, character recognition, and item proposal, to name a few. Artificial intelligence is being used in a variety of applications, including self-driving Google cars, Netflix displaying the films and shows that a person may enjoy, online suggestion engines—such as companion ideas on Facebook, "more things to consider" and "get yourself a little something" on Amazon, and Visa extortion location.

In many cases, human architects design machines that don't perform as well as desired in the circumstances in which they are put to use. To be honest, some characteristics of the working environment may not be fully understood until after the system has been configured. Existing machine designs may be improved by using machine learning techniques.

Long-term trends may shift. Adaptable machines reduce the frequency of machine overhaul by adjusting to changing climatic conditions. People are constantly discovering new pieces of knowledge about their tasks. The language we use has evolved. In our world, there are always fresh occasions to be found. It's not possible to keep updating AI frameworks in response to fresh facts. However, artificial intelligence (AI) methods may be able to keep up with a lot of it in the future.

2.2.2 Classification of ML Techniques

We can use a few areas of AI to address the board's e-mail concerns, and our approach utilized a solo AI strategy to accomplish so. Unsupervised learning is an AI technique in which models from the information space are only provided to the calculation and a model is fitted to these perceptions. Unsupervised learning Bunching calculations are an example of a solo learning activity. Artificial intelligence (AI) models may be fitted to people's perceptions via unaided learning.

Supervised learning is an AI technique for preparing data. Preparation information includes collections of data items (usually vectors) and desired yields. The capacity yield may be a nonstop value (called a relapse) or a class name of the information item (called characterization). The regulated student's task is to estimate the capacity

for each significant information item after seeing many prepared models (for example sets of information and target yield). To do so, the learner must "sensibly" summarize from supplied knowledge to hidden conditions (see inductive predisposition). Comparing and learning alone to enable the calculation to acquire competence with a capacity, the calculation is first provided preparatory information that includes models that include both the inputs and the ideal yields.

Unsupervised learning is a type of machine learning that does not make use of human annotations on data sources like books or newspapers. Using a collection of people-organized models is distinguished from directed learning methods that find out how to carry out an assignment, such as grouping or relapse. This means we are just given the Xs and some (severe) critiques to improve on our presentation when we learn alone. We basically have a collection of vectors for planning purposes only, with no expectation of using them in the future. Normally, the problem in this scenario is to divide the prepared set into smaller groups.

2.2 Review of Segmentation Technique

Active eye detection techniques take advantage of the eye's retro-reflective characteristic and specific lighting to detect the presence of the eye. They are suitable for real-time scenarios in controlled settings, such as eye gaze tracking, iris identification, and other similar techniques [31]. When detecting and tracking eyes, appearance-based techniques do so directly based on their photometric appearance, while template-based methods utilize a generic predesigned eye model to do so. The appearance-based models are constructed directly from the look of the ocular area. Such techniques need a significant amount of data for training classifiers to map the input eye pictures directly to screen coordinates utilizing the form of the eye, the shape of the eyelids, the shape of the eyelashes, as well as the pupil, the iris, and glints [11, 12, 41, 45-46]. In addition, template and eigenspace-based techniques require normalization and are less efficient and time-consuming [27] than other methods. In the template-based techniques, a common eye model is first created, and then the template matching algorithm is used to identify and distort the eyes on the picture to find the best representation of the eye shape. These techniques are accurate in their detection of the eye, but they require the correct initialization of the eye model near the eyes. Moreover, they are computationally costly since they need the use of a high contrast picture [38- 40]. For the diagnosis of RoI, the feature cum shape-based approach requires the identification of certain eye characteristics such as the limbus, pupil, eye corners, or contours, among others. Feature-based techniques often need a lower number of calibration samples than appearance-based methods, which is a significant advantage. The feature-based techniques make use of the features of the eye to identify distinguishing traits such as the limbus, pupil, and corneal reflection surrounding the eyes, among other things. Even though appearance-based approaches are usually more durable than feature-based methods [35,61], they are very sensitive to head motion.

2.3 Review of Depth Estimation Technique

The first neural network model for Depth estimation was proposed by Eigen et al. The model compares the input image with the corresponding depth image obtained by LIDAR scans. In these baseline models only, depth is considered as a parameter for estimating the distance between object and camera. Later replacing UNET encoder architecture with ConvLSTM layer Prof Arun CS Kumar et al. proposed RNN based approach for this problem. With the advancement in the field of machine learning, many models have been proposed which consider other parameters too in estimating the distance. These techniques are mostly administered, requiring ground truth profundity throughout the preparation phase. In any case, this is a check to see whether the settings have changed. As a result, more work is requiring the gathering of more profundity or other remarks, such as known item sizes [36], restricted ordinal profundities [47, 6], directed appearance coordinating with words [42, 43]. Engineered prepared data is an option [31], but a lot of it has changed the appearance and movement of the real world. Several recent investigations have demonstrated that standard construction from movement (SfM) pipelines may provide inadequate preparation signal for both camera posture and depth [45] recently improved profundity estimates by integrating raucous profundity recommendations with traditional sound system computations.

2.4 Gaze Eye Detection

The canny operator is more expensive and time consuming due to complex computations with false zero crossing as compared to Sobel, Prewitt, and Roberts operator but has better performance under noisy conditions. Edge

detection may also be done by using an active contour, which can travel around a picture from its starting condition to a place where it uses the least amount of energy. Edge extraction, picture segmentation, motion tracking, and 3D reconstruction are all common uses in computer vision [28-30]. The blob detection technique takes less time than edge detection but needs a clear background and foreground relationship. It also requires fixed thresholds. However, the selection of thresholds may affect the outcomes significantly as these methods are highly noised sensitive, crucial, and computationally expensive [31-33]. The edge detection techniques can be used to segment any required RoI like glint, eye contours, iris, etc. Different eye gaze models based on glint or iris extraction are discussed below.

2.5 Gaze Model

Zhang et al. developed a new feature fusion approach to iris identification. The robustness of iris identification is improved by extracting global and local iris features [23]. For pupil and iris segmentation, T. Moravcik employs a binary edge map, followed by a quick and reliable circular Hough transform (CHT) method with many parameter changes. The identification of pupils necessitates a variety of iris radii being specified, which consumes additional computing resources. Due to the iris's loss of circular configuration, the anticipated radius must be increased in comparison to the centre location [26]. Using the starburst method, Ryan et al. find pupillary and limbic feature pixels to match a pair of ellipses for iris segmentation. The ICE (iris challenge evaluation) database utilizes IR LEDs to separate the pictures, which are then manually reassembled. There are issues with the database because of the poor contrast between the pupil and the iris. A further problem is that using a basic thresholding method would provide worse results since it fails to partition the pupil correctly. For just 25 people, Sigut et al. used the iris centre corneal reflection (ICCR) method instead of the pupil centre (PCCR) to segment the iris's centre for assessing gaze direction. They used visible light instead of infrared light. The tests made advantage of a maximum resolution of 752 582 pixels. The inclusion of a software component to manage the camera's pan and tilt angles in real time has increased in processing time. Only 70 millimeters have been covered (2.3 feet). Shifting your gaze closer to the display has a significant impact on your calibration [35]. The one-circle method is based on Wang and Sung's utilization of an iris image's ellipse form. To calculate a circle, the researchers used the iris contour as the elliptical edge. One camera is placed on a permanent tripod for head position calculations, while the other is on a pan-tilt unit to follow the eye, as suggested by the new algorithm.

2.6 Object Detection Technique

A single deep neural network can identify objects in images. The SSD method discretizes the bounding box output space into a series of default boxes with varying aspect ratios and scales per feature map location. The organization produces scores and develops acclimations to the container for each item class existing in a default box. The network handles objects of various sizes manually using predictions from various feature maps and resolutions. SSD is better than techniques that depend on object proposition because it removes the age of the proposal and subsequent pixel or component resampling. SSD uses a single network to compute. As a result, SSD is simple to develop and comprehend when used in frameworks that need identity verification. PASCAL experiment results in an object's input image and ground truth boxes are required for training. For training, SSD just needs an input image and some ground truth boxes. For example, the convolutional method assesses a small selection of default boxes with varying perspective proportions in a few component maps. Defined as follows: The default boxes' shapes and confidences should be balanced. During preparation, we compare the default boxes to the truth boxes. A cat and a dog are both regarded as good qualities, whereas others are perceived as flaws.

This is because ground truth data should be assigned to specific yields in the appropriate arrangement of locator yields. To test the SSD work, we used the COCO dataset to train SSD300 and SSD512. We used the COCO dataset to train our SSD300 and SSD512 designs. Because COCO objects are smaller than PASCAL VOC objects, we use smaller default boxes for all levels. Our littlest default box is now 0.15 instead of 0.2, and the default box on conv4 3 is 0.07 instead of 0. We use trainval35k [24] to prepare. We initially train the model for 160k cycles with a 10-3 learning rate, then for 40k cycles with 10-4 and 40k cycles with 10-5. The grouping job for little objects is difficult for SSD without additional component resampling as in Faster R-CNN.

The information increase system portrayed assists with working on the presentation drastically, particularly on little datasets like PASCAL VOC. The arbitrary yields created by the system can be considered as a "zoom in"

activity and can produce numerous bigger preparing models. To execute a "zoom-out" activity that makes more little preparing models, we first arbitrarily place a picture on a material of 16× of the unique picture size loaded up with mean qualities before we do any arbitrary harvest activity. Since we have additional preparation pictures by presenting this new "development" information expansion stunt, we need to twofold the preparation emphasis. We have seen a steady increment of 2%-3% map across various datasets, as displayed in Table 6. In explicit, the new increase stunt altogether works on the presentation on little articles. This outcome highlights the significance of the information expansion procedure for the last model exactness.

2.7 Summary

This chapter provides a review of existing literature available for the gaze estimation model. The examiner stated that researchers concentrated on gaze tracking alone not concentrated much on gaze-depth estimation. Based on this consideration this research developed a gaze-depth estimation model for depth - estimation and classification. In the next chapter research methodology adopted for the proposed MONODEPTH2 is presented.

Chapter 3: Methodology

3.1 Introduction

This research aimed to develop an artificial intelligence model for gaze-depth estimation and classification. To perform the gaze-depth estimation model this research proposed a MONODEPTH2 network for effective gaze-depth estimation and classification. The steps involved in the proposed MONODEPTH2 is presented in this chapter as follows:

3.2 Proposed Method

The proposed MONODEPTH2 model is involved in the construction of an artificial intelligence model for gaze-depth estimation and classification. As the proposed MONODEPTH2 model is performed based on consideration of two scenarios those are:

Gaze-depth estimation

Gaze object classification

The developed model uses monocular object detection and classification. To acquire monocular object gaze - depth estimation and classification of the overall process.

The initial phase incorporates the training phase and is followed by the testing phase. The training is performed with monocular video frame extraction. The examination is based on the consideration of extracted video frames. The steps involved in constructed MONODEPTH2 are presented as follows:

3.2.1 Training Model

We might extract the interpretable profundity from the model by forcing the network to execute image amalgamation using a delegate variable for our scenario profundity or dissimilarity. Given the general posture between those two views, there are an immense number of possible incorrect profundities per pixel that may successfully replicate the smart vision. Typical binocular and multi-see sound system methods resolve this ambiguity by maintaining perfection in the depth maps and determining photograph consistency on patches while determining per-pixel profundity [11]. Like [12, 15, 36], we describe our concern as reducing photometric reprojection errors during preparation. We express the general posture for each source see I_t , regarding the objective picture I_t 's posture, as $T_t \rightarrow t$. We foresee a thick profundity map D_t that limits the photometric reprojection mistake L_p , were

$$L_p = \sum_{t'} pe(I_t, I_{t' \rightarrow t}), \quad (1)$$

$$\text{and } I_{t' \rightarrow t} = I_{t'} \langle proj(D_t, T_{t \rightarrow t'}, K) \rangle. \quad (2)$$

In this case, pe is a photometric recreation blunder, and $project()$ returns the predicted profundities D_t in I_t' . We assume the pre-registered intrinsic K of the multiplicity of viewpoints are identical, although they may be distinct. For our photometric error, we use L1 and SSIM [44] to make our photometric mistake function effectively. We utilize it in [15].

$$pe(I_a, I_b) = \frac{\alpha}{2}(1 - \text{SSIM}(I_a, I_b)) + (1 - \alpha)\|I_a - I_b\|_1,$$

where $\alpha = 0.85$. As in [15] we use edge-aware smoothness

$$L_s = |\partial_x d_t^*| e^{-|\partial_x I_t|} + |\partial_y d_t^*| e^{-|\partial_y I_t|}, \quad (3)$$

where $d * t = dt/dt$ is the mean-standardized opposite profundity from [42] to debilitate contracting of the assessed profundity. In sound system preparation, our source picture I_t is the next to see in the sound system pair to I_t , which takes known relative posture. While relative postures are not known ahead of time for monocular successions, [11] presented that it is feasible to prepare a second posture assessment network to anticipate the overall stances $T_t \rightarrow t'$ utilized in the projection work project. During preparing, we tackle for camera posture and profundity all the while, to limit L_p . For monocular preparing, we utilize the two outlines transiently

neighboring I_t as our source outlines, for example, $I_t \in \{I_{t-1}, I_{t+1}\}$. In blended preparing (MS), I_t incorporates the transiently contiguous edges and the contrary sound system see.

3.2.2 Multi-scale Estimation

Because of the bilinear sampler's slope [21], current models utilize multi-scale profundity expectation and picture reconstruction. The total misfortune is the sum of the individual disasters in the decoder. [12, 15] each decoder layer's photometric error. There are large low-surface areas in the center of the road lower down profundity maps where this tends to create 'openings' (subtleties in the profundity map inaccurately moved from the shading picture). Deeper openings may occur in low-surface regions when the photometric error is ambiguous. This complicates the task for the depth network, now free to predict incorrect depths.

3.3 Construction of Monodepth2

As for our framework, our autoencoder gets a thicker profundity map through five sparsity-invariant convolutional layers. The yield of this module, in particular DD in the figure, is managed by the inward misfortune displayed in the figure and better portrayed in the rest of it. Figure 3.1 shows how the autoencoder is formed: 4 meager convolution layers with diminishing piece size (9, 5, 3, 3), everyone with 16 channels also, step fixed to 1 to keep a similar goal of the information. One last scanty convolution pixel-wise channel is included a request to get a picture that addresses a denser difference map which is utilized for the inward misfortune. Then, at that point, it is connected to the information picture and sent both to the fundamental profundity assessor and to a skirt remaining module that will be additionally examined. During preparing we utilize two autoencoders with shared loads to produce both DDL furthermore, DDR from the left and right scanty ones. Along these lines, we authorize consistency in the misfortunes keeping the entire framework symmetric. The reasoning behind this decision will be examined in the blink of an eye, while removal examinations will feature the commitment presented by such a system. This balance is utilized during preparing just, while at test time a solitary autoencoder measures left meager difference map SDL to produce DDL which is given to the profundity assessor after a link with RGB shading picture.

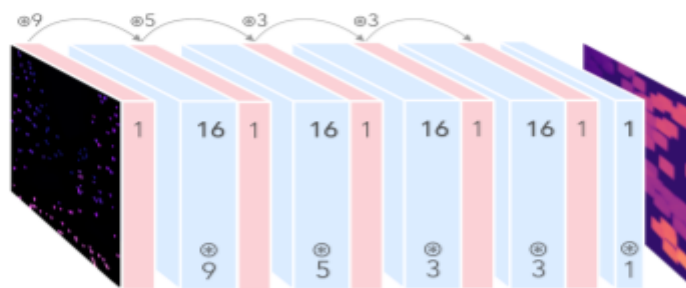


Fig 3.1 Structure of the sparsity-invariant autoencoder

3.4 Dataset

Datasets used here for depth estimation KITTI, NYU, and DDAD. KITTI dataset contains 56 monocular video sequences of both input and output scenarios, the ground truth is calculated using a LIDAR scanner mounted on top of the car. And NYU contains a strong dataset of inside scenarios as monocular video sequences which helps in predicting depth estimation in close range situations or inside situations. The dataset is pre-posed according to the dimensions of the model.

3.5 The SSD model for Object Detection

This method relies on a feed-forward neural network that provides a fixed-size grouping of bounding boxes and scores for the existence of article class events in those compartments, followed by a non-most prominent disguise phase to get the final IDs for the dataset. The early association layers are based on a common plan utilized for breath-taking picture representation (abbreviated before any gathering layers), which we shall refer to as the basic organization in this section. Each new layer of recognition predictions may be generated

by using a collection of convolution channels in a suitable configuration (or a current component layer from the base organization). A 3×3 pad is the basic part for anticipating the boundaries of a prospective revelation for a part layer of dimension $m \times n$ with p stations, and it provides either a score for an order or a shape offset comparable to that of the default box bearings. A bunch of default jumping boxes is related to each component map cell in the organization, which permits us to utilize a few element maps at the highest point of the organization. Due to the convolutional idea of the default boxes, the element map is tiled steadily, with the area of each container comparative with its related cell fixed. We gauge the balances comparative with the default enclose shapes of each element map cell, just as the per-class scores that show the presence of a class example in every one of those crates, for each element map cell. To be more explicit, for each case out of k at a given position, we work out the c class scores just as the four counterbalances from the default box structure, as displayed in the figure. With an $m \times n$ feature map, this leads to an overall total of $(c + 4)k$ filters being applied around each position on the feature map, giving $(c + 4)kmn$ outputs for each location. Please see Fig. 1 for an example of what default boxes look like. The default boxes in Faster R-CNN are like the anchor boxes in Faster R-CNN. We can construct a set of default bounding boxes for each feature map cell, and we can have several feature maps at the top of the network by using the feature map cells. The default constrains the component map to be tiled in a convolutional manner, which means that the relative position of each container concerning its corresponding cell is fixed. The balances relative to the default confine forms the cell, as well as the per-class scores that demonstrate the existence of a class case in each of those crates, are anticipated at every component map cell. We analyse c class scores and the four counterbalances for each container out of k in each region, and we compare the results to the initial default box shape. Thus, for an $m \times n$ include map, an aggregate of $(c + 4)k$ channels are applied around each region in the component map, giving an aggregate of $(c + 4)kmn$ yields for each area in the component map.

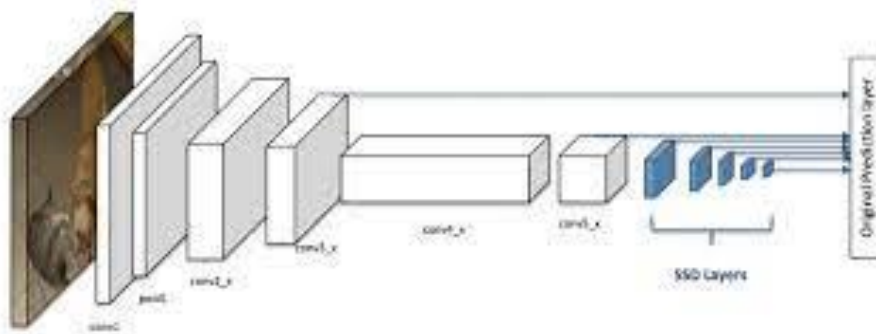


Fig 3.2 Architecture of SSD Model

These models are integrated together for finding the gaze-depth estimation and classification. The proposed model utilizes coordinates obtained using Pupil labs software and these values are fed to DARKNET integrated with YOLO/SSD for classification and Monodepth2 for depth estimation of that particular pixel. The extracted and processed frames of monocular video are created as a database. The created monocular dataset consists of training data with the extracted video frame. The dataset for a monocular video frame is extracted and processed based on consideration of the following parameters as follows:

1. Object detection from monocular sequences
2. Estimation of the number of objects lies within the frame
3. Measuring the object distance based on depth estimation
4. Inference of camera at the time instances.

3.6 Summary

This chapter presented a proposed methodology adopted for gaze-depth estimation and classification. The proposed MONODEPTH2 incorporates training on the architecture for gaze-depth estimation and classification. The proposed MONODEPTH2 is implemented in the Python platform and simulation results are presented in the next chapter.

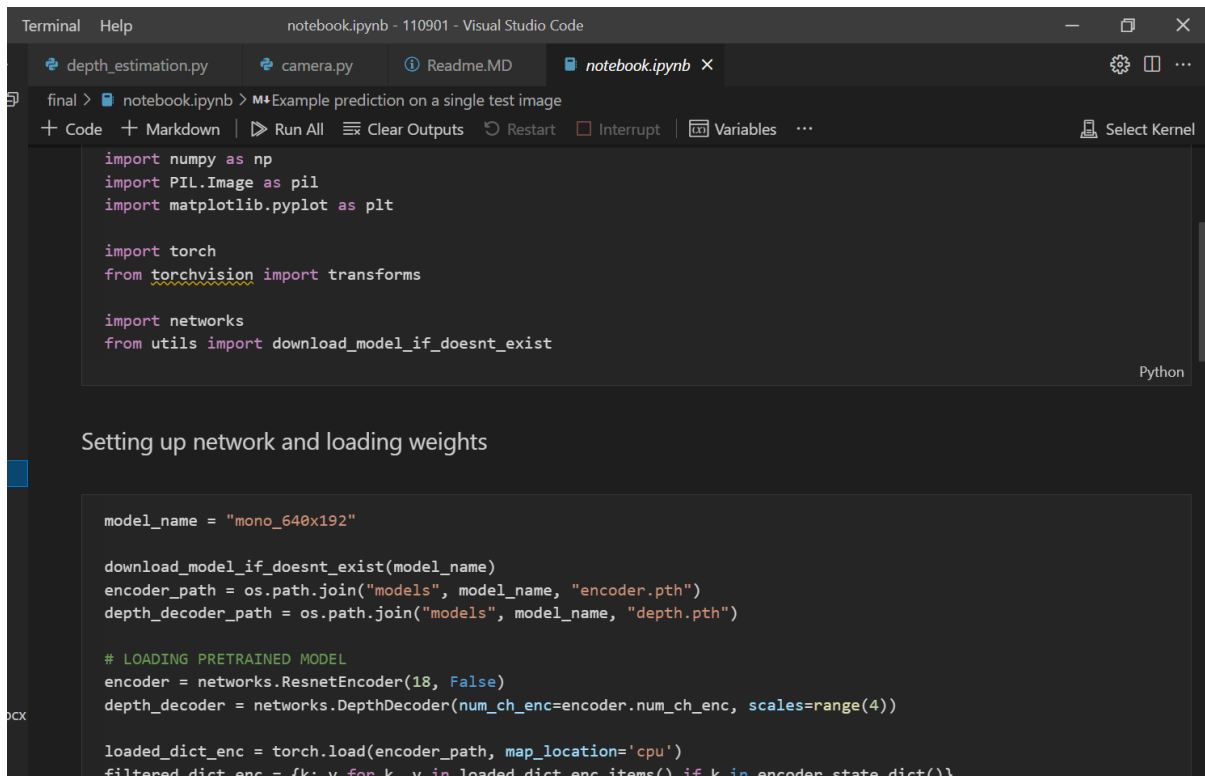
Chapter 4: Result and Discussion

4.1 Introduction

This research developed an artificial gaze-depth estimation and classification model for classification. The developed artificial model incorporates regression based. This research utilizes the monocular video sequences for the extraction of the video frame. The proposed MONODEPTH2 incorporates a regression-based classifier model for gaze-depth estimation and classification. The performance of the proposed MONODEPTH2 is evaluated in simulation software Python. The implemented parameters are described in this chapter.

4.2 Simulation Code

I implemented the MONODEPTH2 model is download as zip format and placed or located in the model's folder inside the codes folder. In figure 4.1 (a) I am showing you the screenshot of the notebook opened in vs code and contained the evaluation part of code for modepth2 model on the single images of any size. I also used it for multiple images evaluation. The result or depth graph of single image output is shown in figure 4.1 (b).



```
Terminal  Help  notebook.ipynb - 110901 - Visual Studio Code

depth_estimation.py  camera.py  Readme.MD  notebook.ipynb X

final > notebook.ipynb > Example prediction on a single test image
+ Code + Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables | Select Kernel

import numpy as np
import PIL.Image as pil
import matplotlib.pyplot as plt

import torch
from torchvision import transforms

import networks
from utils import download_model_if_doesnt_exist

Python

Setting up network and loading weights

model_name = "mono_640x192"

download_model_if_doesnt_exist(model_name)
encoder_path = os.path.join("models", model_name, "encoder.pth")
depth_decoder_path = os.path.join("models", model_name, "depth.pth")

# LOADING PRETRAINED MODEL
encoder = networks.ResnetEncoder(18, False)
depth_decoder = networks.DepthDecoder(num_ch_enc=encoder.num_ch_enc, scales=range(4))

loaded_dict_enc = torch.load(encoder_path, map_location='cpu')
filtered_dict_enc = {k: v for k, v in loaded_dict_enc.items() if k in encoder.state_dict()}
```

Fig 4.1 (a) Loading Monodepth model



Fig 4.1 (b) Depth map

The above image is given input and the below image is obtained output, the brighter the pixel, the closer the approaching vehicle. Since the model is trained on ground truths obtained by LIDAR scans, the scanner outputs high intensity value based on the pixel distance closer to the viewer, so our model assigns high intensity pixel values to the closest obstacle.

The presented figure 4.1 c. shown the code for object detection part and the implementation done on the SSD model. This code contains in code.py, it is also evaluation code but this time for multiple images.

```

Run Terminal Help          eval.py - obj_detection - Visual Studio Code
...  obj_det_model.py  eval.py  X  obj_det_data_loader.py  utils.py
eval.py
1  import torch
2  from obj_det_model import ssd_model
3  from obj_det_model import utils
4  from PIL import Image
5  from torchvision import transforms
6  from utils import get_iou
7  |
8
9  device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
10 ssd_model.to(device)
11 ssd_model.eval()
12
13 # evaluate single images
14 def eval_img(tensor):
15     with torch.no_grad():
16         detections_batch = ssd_model(tensor)
17         results_per_input = utils.decode_results(detections_batch)
18         best_results_per_input = [utils.pick_best(results, 0.25) for results in results_per_input]
19         return best_results_per_input
20
21 # convert image to tensor
22 def img_loader(img_path):
23     image = Image.open(img_path).resize((300,300)).convert('RGB')
24     loader = transforms.Compose([transforms.ToTensor()])
25     image = loader(image).unsqueeze(0)
26     return image.to(device, torch.float)
27
28 # calculate matrix for testing
29 def calc_player_detection_metrics(x, y, w, h, count):
30     # d = read_annotations('/drive/Shared with me/InTERNship/DetResults/annotations', count)

```

Fig 4.1(c) Object Detection evaluation

4.3 Constructed Model

The training modality of the monocular data were presented as persons, car, trucks and so on. Those pixels size was varies from 640 x 192 to 1024 x 320.

```

code.ipynb X
notebooks > code.ipynb > # import libraries
+ Code + Markdown | Run All | Clear Outputs | Restart | Interrupt | Variables ... Python 3.9.5 64-bit
# for evaluating model
def evaluate_model(n, root_dir):
    tp, fp, fn = 0,0,0
    for i in range(1,n):
        img_name = 'IMG_9851_frame_'+ str(i).rjust(6,'0')+ '.JPG'
        img_path = root_dir+ '/' + img_name
        img_tensor = img_loader(img_path)
        best_results = eval_img(img_tensor)
        for image_idx in range(len(best_results)):
            bboxes, classes, confidences = best_results[image_idx]

            show_img(img_path, bboxes, classes, confidences, i)

        for idx in range(len(bboxes)):
            left, bot, right, top = bboxes[idx]
            x, y, w, h = [val * 300 for val in [left, bot, right - left, top - bot]]

            # if classes[idx] -1 == 0:
            #     annotate_img(img_path, x, y, w, h, (i,idx))
            #     crop_and_save(img_path, x, y, w, h, (i,idx))
            #     print(x,y,w,h, confidences[idx])
            pred = calc_player_detection_metrics(x, y, w, h, (i, idx))
            if pred == 'tp':
                tp += 1
            elif pred == 'fp':
                fp += 1
            else: fn += 1

```

Fig 4.2 Object Detection evaluation

The above model shows the testing part of the code for the object detection model, the function `evaluate_model` that is shown in figure 4.2 is for the testing of the custom dataset of images for object detection with a size of 300 by 300 pixels.

4.4 Simulation Results

4.4.1 Object Detection Results

The training and testing losses of object detection with a custom dataset with an SSD model implemented with python. The loss and accuracy graphs are shown below.

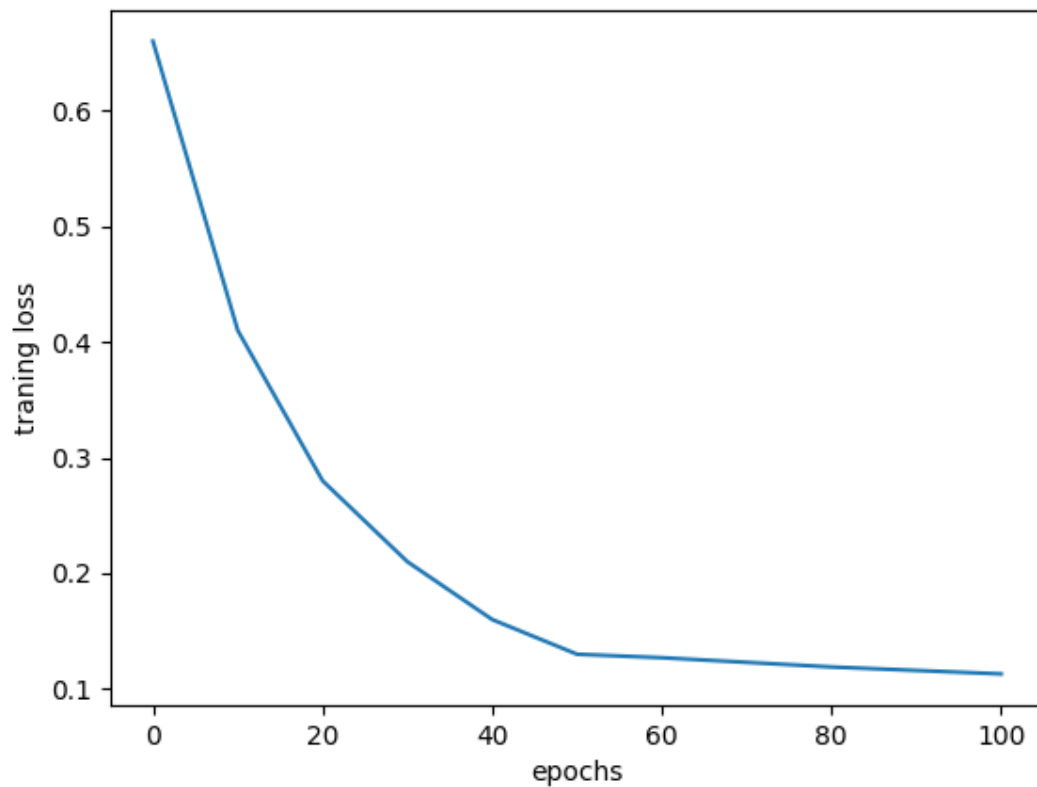


Fig 4.3 Training loss with 100 epochs.

Figure 4.3 represents the training loss graph for the object detection with SSD model with Cifar 10 dataset the final loss obtained after 100 epochs is 0.115.

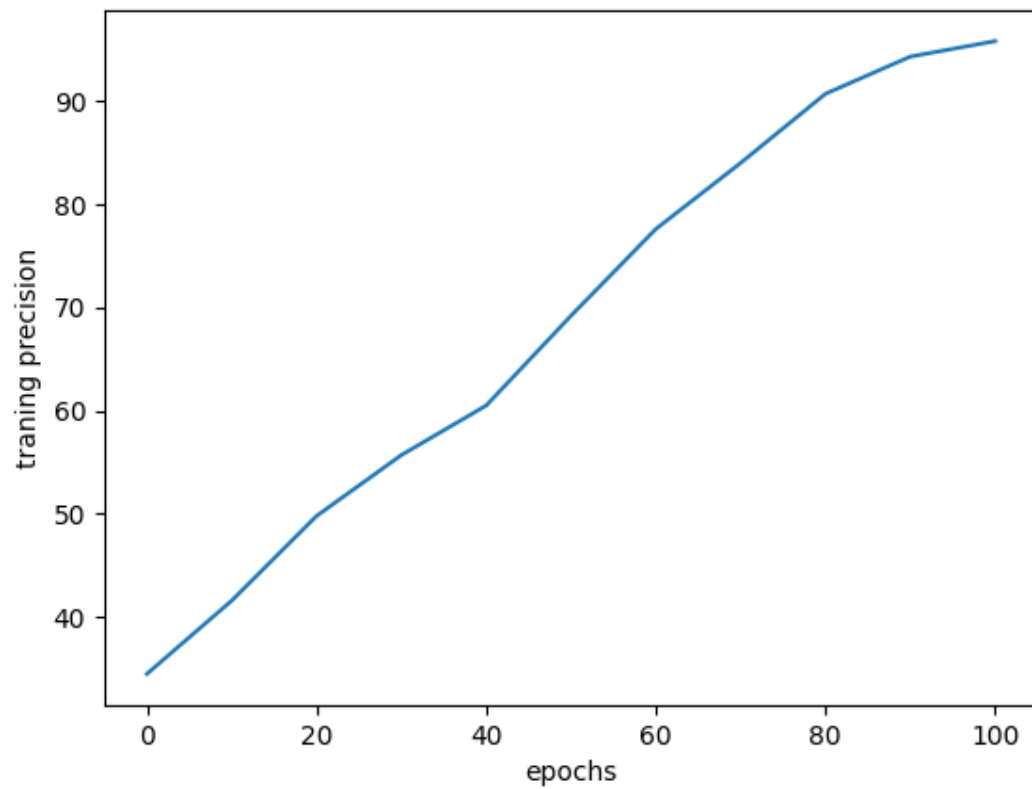


Fig 4.4 Training Accuracy with 100 epochs.

Figure 4.4 represents the training accuracy graph for the object detection with SSD model with Cifar 10 dataset the final accuracy obtained after 100 epochs is 96.43 %.

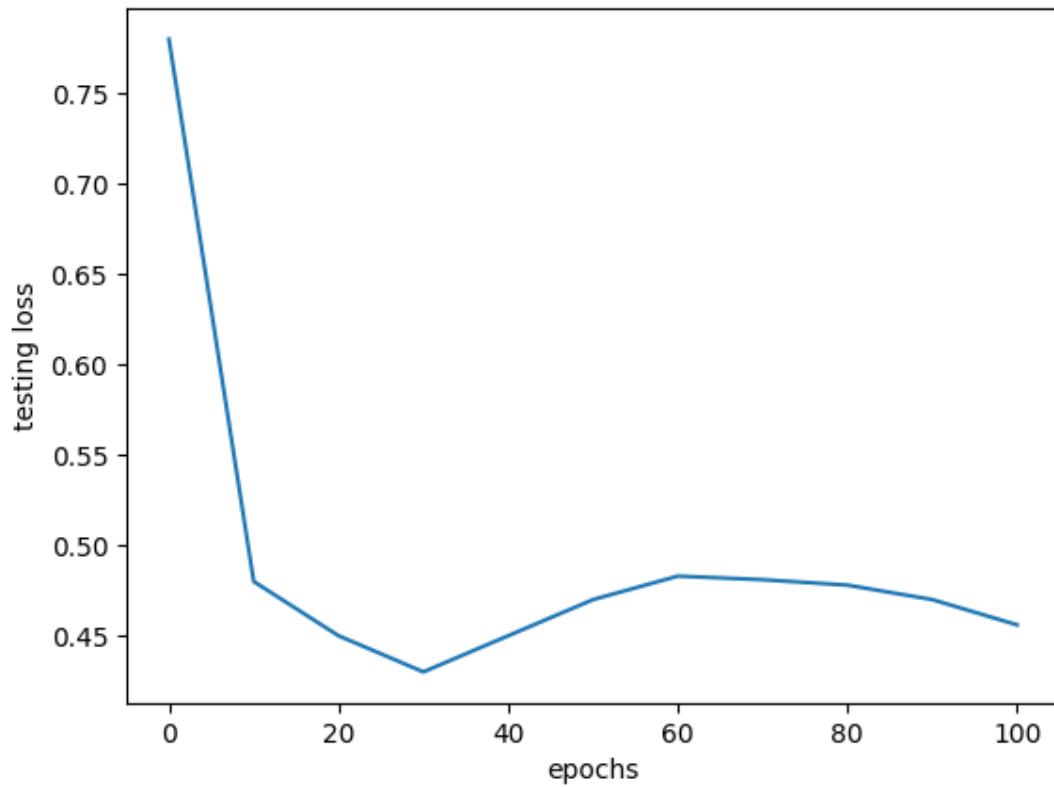


Fig 4.5 Testing Loss with 100 epochs.

Figure 4.5 represents the testing loss graph for the object detection with the SSD model with Cifer 10 dataset the final loss obtained after 100 epochs are 0.459 but the lowest loss is 0.215.

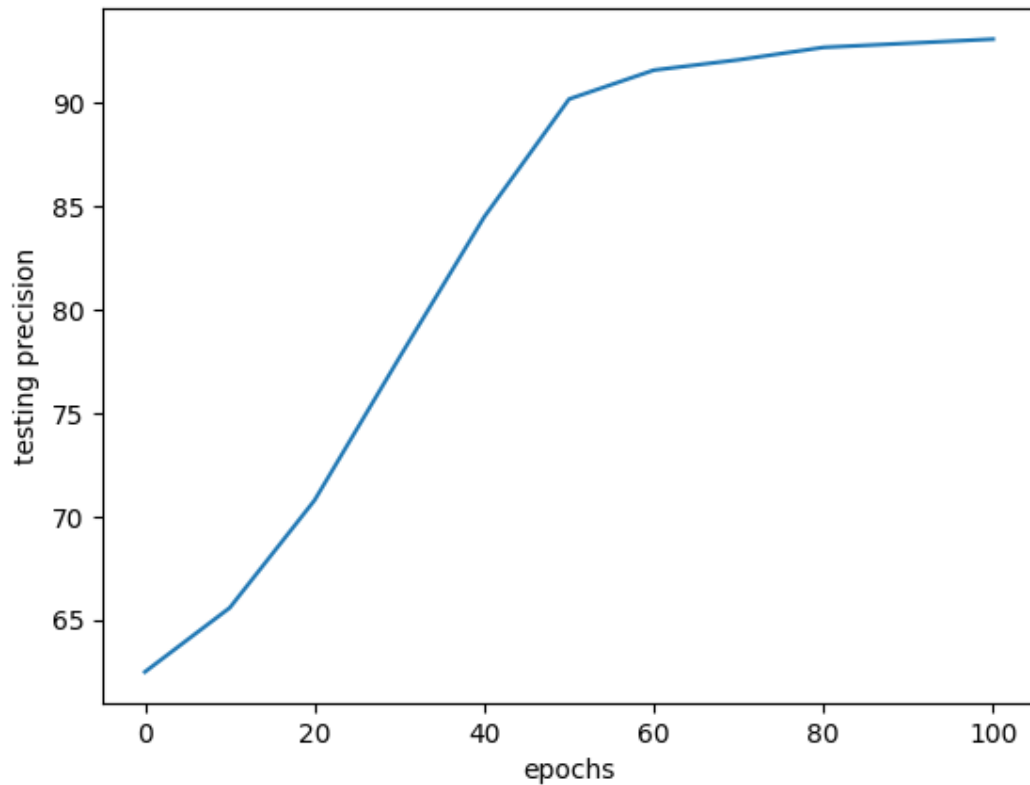


Fig 4.6 Testing Accuracy with 100 epochs.

Figure 4.3 represents the testing precision graph for the object detection with SSD model with Cifar 10 dataset the final accuracy obtained after 100 epochs are 95.8%.

+ Code + Text				RAM	Disk	Editing
Epoch 24	Training loss 0.24874219769978767	Testing loss 0.46128249339237337	Training Accuracy 0.914	Testing Accuracy 0.857		
Epoch 25	Training loss 0.24359598816336725	Testing loss 0.4502178153414635	Training Accuracy 0.917	Testing Accuracy 0.859		
Epoch 26	Training loss 0.23725251386613797	Testing loss 0.4584440937277618	Training Accuracy 0.917	Testing Accuracy 0.859		
Epoch 27	Training loss 0.23743804663961577	Testing loss 0.47238443611533776	Training Accuracy 0.918	Testing Accuracy 0.85		
Epoch 28	Training loss 0.23232848417939074	Testing loss 0.46730658430962047	Training Accuracy 0.92	Testing Accuracy 0.854		
Epoch 29	Training loss 0.2270311392043405	Testing loss 0.48867239297043746	Training Accuracy 0.922	Testing Accuracy 0.853		
Epoch 30	Training loss 0.21134247266880388	Testing loss 0.43920933735218776	Training Accuracy 0.929	Testing Accuracy 0.866		
Epoch 31	Training loss 0.1996156746030921	Testing loss 0.44142621983388425	Training Accuracy 0.931	Testing Accuracy 0.865		
Epoch 32	Training loss 0.19574938246222864	Testing loss 0.4468683180440763	Training Accuracy 0.933	Testing Accuracy 0.864		
Epoch 33	Training loss 0.18708552794573863	Testing loss 0.45303457614722525	Training Accuracy 0.937	Testing Accuracy 0.862		
Epoch 34	Training loss 0.18612097410480385	Testing loss 0.4522989584002525	Training Accuracy 0.937	Testing Accuracy 0.862		
Epoch 35	Training loss 0.1806815586926992	Testing loss 0.46248397202628433	Training Accuracy 0.939	Testing Accuracy 0.86		
Epoch 36	Training loss 0.18081519365444054	Testing loss 0.4638922987090554	Training Accuracy 0.939	Testing Accuracy 0.861		
Epoch 37	Training loss 0.17494308875784186	Testing loss 0.4618172296293222	Training Accuracy 0.94	Testing Accuracy 0.86		
Epoch 38	Training loss 0.17482424684612038	Testing loss 0.4712552691151382	Training Accuracy 0.941	Testing Accuracy 0.86		
Epoch 39	Training loss 0.17294614105616385	Testing loss 0.46877259063492915	Training Accuracy 0.94	Testing Accuracy 0.86		
Epoch 40	Training loss 0.16309269609125068	Testing loss 0.45667370608088315	Training Accuracy 0.946	Testing Accuracy 0.862		
Epoch 41	Training loss 0.1537516049426192	Testing loss 0.46533747957010935	Training Accuracy 0.949	Testing Accuracy 0.863		
Epoch 42	Training loss 0.15126143336353248	Testing loss 0.461428447703647	Training Accuracy 0.95	Testing Accuracy 0.865		
Epoch 43	Training loss 0.15594470887294066	Testing loss 0.467649697118504	Training Accuracy 0.948	Testing Accuracy 0.863		
Epoch 44	Training loss 0.14895211672529463	Testing loss 0.4701360057873331	Training Accuracy 0.951	Testing Accuracy 0.864		
Epoch 45	Training loss 0.1501765311088251	Testing loss 0.46835838305722377	Training Accuracy 0.951	Testing Accuracy 0.862		
Epoch 46	Training loss 0.1470915866929971	Testing loss 0.4702462092706352	Training Accuracy 0.951	Testing Accuracy 0.862		
Epoch 47	Training loss 0.146558639319504	Testing loss 0.46794220843132894	Training Accuracy 0.952	Testing Accuracy 0.862		
Epoch 48	Training loss 0.14561756388486727	Testing loss 0.47310294120744534	Training Accuracy 0.951	Testing Accuracy 0.861		
Epoch 49	Training loss 0.14355944245910782	Testing loss 0.4769854253264749	Training Accuracy 0.953	Testing Accuracy 0.862		
Epoch 50	Training loss 0.13894263392938372	Testing loss 0.47334054726988645	Training Accuracy 0.955	Testing Accuracy 0.862		
Epoch 51	Training loss 0.13774048198669246	Testing loss 0.4747332580831322	Training Accuracy 0.954	Testing Accuracy 0.864		
21s completed at 1:20 PM						

Fig 4.7 Training on Google Collab.

Result of Depth Estimation Mode:

Model	RMSE	RMSE log
ConvLSTM(proposed)	5.649	0.239
Eigen et al	6.563	0.292
Godard et al	5.927	0.247

Table 1. Comparison between baseline models for depth estimation

The ConvLSTM can predict accurately the depth map for a given input monocular sequence and is top performing among the baseline models, but it fails when compared with state of art models, as only depth is considered for solving our problem.

Model	Abs Rel	Sq Rel	RMSE	RMSElog	Delta < 1.25	Delta < 1.25^2	Delta < 1.25^3
VNL	0.072	0.883	3.258	0.117	0.938	0.990	0.998
PackNet-SfM	0.078	0.420	3.485	0.121	0.931	0.986	0.996
DeepV2d	0.037	0.174	2.005	0.074	0.977	0.993	0.997
Monodepth	0.091	0.548	3.79	0.181	0.892	0.956	0.979
Monodepth2(trained)	0.097	0.891	4.863	0.193	0.877	0.959	0.981

Table 2 Comparison between state of art models for depth estimation

The training modality of the monocular data was presented as persons, cars, trucks, and so on. Those pixel sizes were varied from 640 x 192 to 1024 x 320. The input monocular dataset was processed and evaluated to improve the performance. Based on the training model the evaluation is based on the estimation and classification. The proposed GDE involved in-depth - estimation and classification. Based on the video sequences the proposed GDE computes the depth, number of frames processed per second and objects..

```

IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
person: 0.615886
person is at 9 Inches
Gaze Distance: 11.72
Gaze Depth. FPS: 0.1
Gaze Distance: 4.16
Gaze Depth. FPS: 0.2
Gaze Distance: 2.59
Gaze Depth. FPS: 0.4
Gaze Distance: 1.84
Gaze Depth. FPS: 0.5
person: 0.727019
person is at 9 Inches
Gaze Distance: 2.62
Gaze Depth. FPS: 0.4
person: 0.851112
person is at 8 Inches
Gaze Distance: 2.57
Gaze Depth. FPS: 0.4
person: 0.892217
person is at 8 Inches
Gaze Distance: 2.25
Gaze Depth. FPS: 0.4
person: 0.870216
person is at 8 Inches
Gaze Distance: 2.70
Gaze Depth. FPS: 0.4
Gaze Distance: 2.45
Gaze Depth. FPS: 0.4
Gaze Distance: 2.35
Gaze Depth. FPS: 0.4

```

Fig 4.8 Gazed object depth estimation

The above screen provides the output obtained for provided input. The output provides the number of a frame processed, gaze -depth and distance between objects are estimated. In the below table presented, values obtained for sequences are presented.

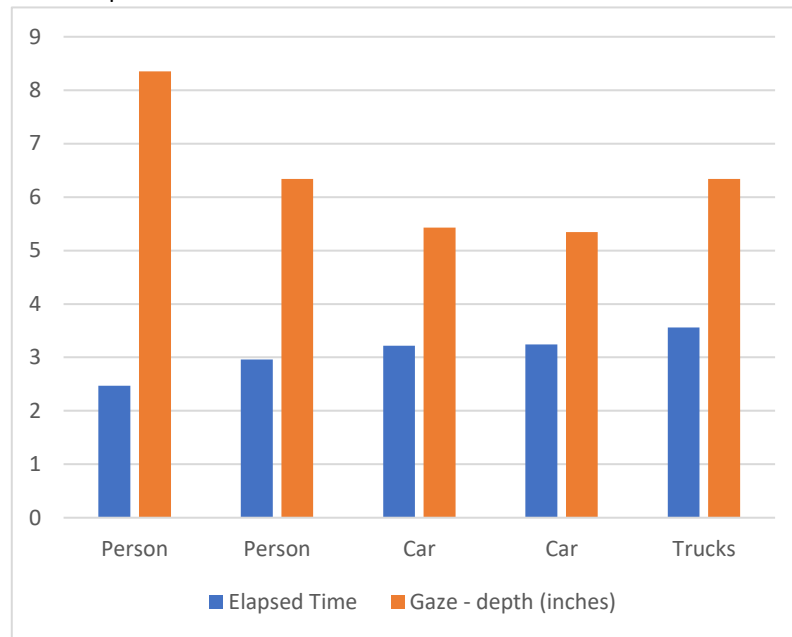


Fig 4.9 Elapsed time graph

4.5 Summary

This chapter presented a result obtained for the proposed GDE for gaze-depth estimation and classification. The proposed model utilizes the monocular video sequences for frame gaze-depth estimation and classification. The analysis stated that the proposed model significantly performs gaze-depth estimation and classification. The overall conclusion of the proposed GDE is presented in the next chapter.

Chapter 5: Conclusion and Future works

After completion of this task of gaze-depth estimation and object detection, here in this section, I am going to conclude all the important points that need to be considered in this research paper, first of all, this process that followed in this methodology is based on the two-step process, first one is gaze depth estimation and second one object detection. Once training is done the models are integrated together for our application.

For gaze-depth estimation, I used the Monodepth2 model architecture and ConvLSTM architecture shown in the upper part and these models are trained on KITTI dataset. For the Object detection part, I used the SSD model architecture, and it is trained on the CIFAR-100 dataset. Overall results are compared in the table below, it contains the comparison of both object detection and gaze-depth estimation model comparison with state of art algorithms.

Table 5.1 Evaluation result for KITTI dataset of different models.

Model	Abs Rel	Sq Rel	RMSE	RMSElog	Delta < 1.25	Delta < 1.25 ²	Delta < 1.25 ³
VNL	0.072	0.883	3.258	0.117	0.938	0.990	0.998
PackNet-SfM	0.078	0.420	3.485	0.121	0.931	0.986	0.996
DeepV2d	0.037	0.174	2.005	0.074	0.977	0.993	0.997
Monodepth	0.091	0.548	3.79	0.181	0.892	0.956	0.979
Monodepth2(trained)	0.097	0.891	4.863	0.193	0.877	0.959	0.981

As clearly shown in Table 5.1 the RMSE log value is lowest for DeepV2d Model but for monodepth2 model, due to compactible issues, pre-trained monodepth2 was considered. And it is easy to implement for training and production. For the object detection part, the SSD and YOLO are fast and efficient. The integrated model gave fruitful results that can be utilized for Depth-gaze analysis, apart from my purpose (Navigation).

In the proposed architecture, the gaze system isn't integrated with the model due to high training time and model complexity and just the co-ordinate points have been used and forced to depend on estimated co-ordinate points, in future works, a model architecture that includes gaze system is more reliable.

References

1. Lupu, R. G., Ungureanu, F., & Bozomitu, R. G. (2012). Mobile embedded system for human computer communication in assistive technology. 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing, 209–212. Cluj-Napoca, Romania, pp. 209-212, August 2012.
2. D.Eigen,C.Puhrsch and R.Fergus (2014). Depth map prediction from a Single image using a Multi-Scale Deep Network.
3. Moldoveanu, A. D., Moldoveanu, F., Asavei, V., Egner, A., & Morar, A. (2013). From HTML to 3DMMO – A roadmap full of challenges. Advances in Intelligent Systems and Computing Advances in Intelligent Control Systems and Computer Science, 379–392.
4. Lalonde, M., Beaulieu, M., & Gagnon, L. (2001). Fast and robust optic disc detection using pyramidal decomposition and Hausdorff-based template matching. IEEE Transactions on Medical Imaging, 20, 1193–1200.
5. Kauppi, T., Kalesnykiene, V., Kamarainen, J. K., Lensu, L., Sorri, I., Uusitalo, H., Kalviainen, H., & Pietila, J. (2006). Diaretdb0: Evaluation database and methodology for diabetic retinopathy algorithms. Technical report, Finland: Lappeenranta University of Technology.
6. Dehghani, A., Moghaddam, H. A., & Moin, M. (2012). Optic disc localization in retinal images using histogram matching. EURASIP Journal on Image and Video Processing, 2012, 19.
7. Duygulu, P., Barnard, K., Freitas, J. F., & Forsyth, D. A. (2002). Object recognition as machine translation: Learning a Lexicon for a fixed image vocabulary. Computer Vision – ECCV 2002 Lecture Notes in Computer Science, Vol. 2353, Germany, pp. 97–112.
8. Blanco, M., Penedo, M. G., Barreira, N., Penas, M., & Carreira, M. J. (2006). Localization and extraction of the optic disc using the fuzzy circular Hough transform. Artificial Intelligence and Soft Computing – ICAISC 2006 Lecture Notes in Computer Science, 712–721
9. Budai, Attila, Bock, Rüdiger, Maier, Andreas, Hornegger, Joachim, Michelson, Georg. Robust Vessel Segmentation in Fundus Images. International Journal of Biomedical Imaging, vol. 2013, 2013.
10. Aggarwal, M. K., & Khare, V. (2015). Automatic localization and contour detection of optic disc. 2015 International Conference on Signal Processing and Communication (ICSC), Noida, pp. 406–409.
11. Akhade, S. B., Deshmukh, V. U., & Deosarkar, S. B. (2014). Automatic optic disc detection in digital fundus images using image processing techniques. International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, pp. 1–5.
12. Balan, O., Moldoveanu, A., Moldoveanu, F., Morar, A., Asavei, V. (2013). Assistive IT for visually impaired people. Journal of Information Systems & Operations Management, 7, 391–403.
13. Akyol, K., Şen, B., & Bayır, Ş. (2016). Automatic Detection of Optic Disc in Retinal Image by Using Keypoint Detection, Texture Analysis, and Visual Dictionary Techniques. Computational and Mathematical Methods in Medicine, 2016, 1-10.
14. Gutte, V. D., & Vaidya, Y. M. (2014). Detection of optic disc and cup of digital fundus image using salient object detection method for glaucoma detection. Proceedings of the 2nd International Conference on Intelligent Systems and Image Processing 2014.
15. Jampani, V. U., Sivaswamy, J., & Vaidya, V. (2012). Assessment of computational visual attention models on medical images. Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing – ICVGIP 12. Mumbai, India-December 16 - 19, 2012, p. 80-88.
16. Jarodzka, H., & Brand-Gruwel, S. (2017). Tracking the reading eye: Towards a model of real-world reading. Journal of Computer Assisted Learning, 33, 193–201
17. Jarodzka, H., Janssen, N., Kirschner, P. A., & Erkens, G. (2015). Avoiding split attention in computer-based testing: Is neglecting additional information facilitative? British Journal of Educational Technology, 46, 803–817.
18. Lee, E. C., Woo, J. C., Kim, J. H., Whang, M., & Park, K. R. (2010). A brain– computer interface method combined with eye tracking for 3D interaction. Journal of Neuroscience Methods, 190, 289–298.
19. Ma, K., Sim, T., & Kankanhalli, M. (2013). VIP: A unifying framework for computational eye-gaze research. Human Behavior Understanding Lecture Notes in Computer Science, 209–222.

20. Mason, L., Tornatora, M. C., & Pluchino, P. (2013). Do fourth graders integrate text and picture in processing and learning from an illustrated science text? Evidence from eye-movement patterns. *Computers & Education*, 60), 95–109.
21. Mendonça, A. M., Sousa, A., Mendonça, L., & Campilho, A. (2013). Automatic localization of the optic disc by combining vascular and intensity information. *Computerized Medical Imaging and Graphics*, 37, 409–417.
22. Niemeijer, M., Xu, X., Dumitrescu, A. V., Gupta, P., Ginneken, B. V., Folk, J. C., & Abramoff, M. D. (2011). Automated measurement of the arteriolar-to-venular width ratio in digital colour fundus photographs. *IEEE Transactions on Medical Imaging*, 30, 1941–1950.
23. Pasupa, K., Saunders, C. J., Szedmak, S., Klami, A., Kaski, S., & Gunn, S. R. (2009). Learning to rank images from eye movements. 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 27 Sept.-4 Oct. 2009, Kyoto, Japan, pp. 2009–2016, 2009..
24. Reza, M. N., & Ahmad, M. (2015). Automatic detection of optic disc in fundus images by curve operator. 2015 2nd International Conference on Electrical Information and Communication Technologies (EICT), 10-12 Dec. 2015, Khulna, Bangladesh, pp. 143-147, 2015
25. Salazar-Gonzalez, A., Kaba, D., Li, Y., Liu, X. (2014). Segmentation of the blood vessels and optic disk in retinal images. *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 6.
26. Santella, A., Agrawala, M., Decarlo, D., Salesin, D., & Cohen, M. (2006). Gazebased interaction for semi-automatic photo cropping. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems – CHI 06*, Montréal, Québec, Canada — April 22 - 27, pp. 771-780, 2006.
27. Sopharak, A., Uyyanonvara, B., Barman, S., & Williamson, T. H. (2008). Automatic detection of diabetic retinopathy exudates from non-dilated retinal images using mathematical morphology methods. *Computerized Medical Imaging and Graphics*, 32, 720–727.
28. Tjandrasa, H., Wijayanti, A., & Suciati, N. (2012). Optic nerve head segmentation using Hough transform and active contours. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 10, 531.
29. Vidal, M., Turner, J., Bulling, A., & Gellersen, H. (2012). Wearable eye tracking for mental health monitoring. *Computer Communications*, 35, 1306–1311.
30. Vilariño, F., Lacey, G., Zhou, J., Mulcahy, H., & Patchett, S. (2007). Automatic labeling of colonoscopy video for cancer detection. *Pattern Recognition and Image Analysis Lecture Notes in Computer Science*, 290–297.
31. Wisaeng, K., Hiransakolwong, N., & Pothiruk, E. (2014). Automatic detection of optic disc in digital retinal images. *International Journal of Computer Applications*, 90, 15–20.
32. Yang, Y., & Wang, C. (2015). Trend of using eye tracking technology in business research. *Journal of Economics, Business and Management*, 3, 447–451.
33. Zhang, K., Zhang, L., Song, H., & Zhou, W. (2010). Active contours with selective local or global segmentation: A new formulation and level set method. *Image and Vision Computing*, 28, 668–676.
34. V. Arkorful and N. Abaidoo, “The role of e-learning, advantages and disadvantages of its adoption in higher education,” *International Journal of Instructional Technology and Distance Learning*, vol. 12, no. 1, pp. 29–42, 2015
35. R. Taylor, “The multimodal texture of engagement: Prosodic language, gaze and posture in engaged, creative classroom interaction,” *Thinking Skills and Creativity*, vol. 20, pp. 83–96, 2016.
36. K. Sharma, P. Jermann, and P. Dillenbourg, “with-me-ness: A gaze-measure for students’ attention in moocs,” in *Proceedings of International Conference of the Learning Sciences 2014*, no. EPFL-CONF-201918, pp. 1017–1022, ISLS, 2014.
37. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New York, New York: Springer Science and Business Media.
38. Forsyth, R. S. (1990). The strange story of the Perceptron. *Artificial Intelligence Review*, 4 (2), 147-155.
39. Haykin, S. (19994). *Neural Networks: A Comprehensive Foundation*. New York: Macmillan Publishing.
40. Oltean, M. (2005). Evolving Evolutionary Algorithms Using Linear Genetic Programming. 13 (3), 387 - 410 .
41. Orlitsky, A., Santhanam, N., Viswanathan, K., & Zhang, J. (2005). Convergence of profile based estimators. *Proceedings of International Symposium on Information Theory. Proceedings. International Symposium on*, pp. 1843 - 1847. Adelaide, Australia: IEEE.

42. Tapas Kanungo, D. M. (2002). A local search approximation algorithm for k-means clustering. Proceedings of the eighteenth annual symposium on Computational geometry (pp. 10-18). Barcelona, Spain : ACM Press.
43. Hörl, S., Becker, F., Dubernet, T., Axhausen, K.W., 2019. Induzierter Verkehr durch autonome Fahrzeuge: Eine Abschätzung. Technical Report. Research project SVI 2016/001 upon request of the Swiss Association of Transport Engineers and Experts (SVI)
44. Loeb, B., Kockelman, K.M., 2019. Fleet performance and cost evaluation of a shared autonomous electric vehicle (SAEV) fleet: A case study for Austin, Texas. Transp. Res. Part A: Policy Pract. 121, 374–385
45. Wadud, Z., 2017. Fully automated vehicles: A cost of ownership analysis to inform early adoption. Transp. Res. Part A: Policy Pract. 101, 163–176. <https://doi.org/10.1016/j.tra.2017.05.005>
46. 3 Y. Hübner, P. T. Blythe, G. Hill, M. Neaimeh, J. Austin, L. Gray, et al., “49,999 Electric car journeys and counting,” presented at the EVS27, Barcelona, Spain, 2013 by Josey Wardle
47. O Martin, E., Shaheen, S., (2016) The Impacts of Car2go on Vehicle Ownership, Modal Shift, Vehicle Miles Travelled, and Greenhouse Gas Emissions: An analysis of five North American cities, Transportation Sustainability Research Center (TSRC), UC Berkeley. Accessed online, available at: http://innovativemobility.org/wp-content/uploads/2016/07/Impactsofcar2go_FiveCities_2016.pdf
48. European Commission (2015), Commission Implementing Decision of 15.12.2015 on a standardisation request to the European Committee for Standardisation, to the European Committee for Electrotechnical Standardisation and to the European Telecommunications Standards Institute pursuant to Regulation (EU) No 1025/2012 of the European Parliament and of the Council as regards certain measuring instruments. EC, Brussels
49. Competition and Markets Authority (2016) Energy market Investigation, Final report, 24 June 2016, CMA, Accessed July 2016, Available at: https://assets.publishing.service.gov.uk/media/5773de34e5274a0da3000113/final_report_energy-market-investigation.pdf
50. European Commission (2016) Air Pollutants from Transport, European Commission.

Appendix: GitLab Repository

The SSD model uses the Cifar100 Dataset for detection.

How to use it

The information can be downloaded here: train+val (257 GB, md5 checksum: c0da97967f76da80f86d6f97d0d98904). To stack the dataset, if it's not too much trouble, utilize the TRI Dataset Governance Policy (DGP) codebase.

The DGP codebase gives various capacities that permit stacking one or different camera pictures, projecting the lidar point cloud into the camera pictures, intrinsic, and extrinsic support, and so forth. Moreover, if it's not too much trouble, allude to the Packnet-SfM codebase (in PyTorch) for additional subtleties on the best way to coordinate and utilize DDAD for profundity assessment preparing/derivation/assessment and cutting edge pretrained models.

For Object detection, you can run either the train.py from object-det or use the code notebook from the notebook section.