Software Requirements Specification for Monocular Depth Estimation Model

Prepared by: Group 11

Krishnapriya Rajeev	B39	krishofficial05@gmail.com
Namitha S Vijayan	B48	namithasvijayan@gmail.com
Saharsh Santhosh Babu	B62	saharshsbabu@gmail.com
Aravind S R	B71	200269@tkmce.ac.in

Table of Contents

Revision History

Name	Date	Reason for Change	Version
	13-03-2023	Improving layout of the document as well as updating product overview, functionality, and constraints	2.0
	25-04-2023	Appendix modified	3.0

1. Introduction

1.1 Document Purpose

The objective of this Software Requirements Specification is to offer a well-defined and documented representation of the requirements needed for the Monocular Depth Estimation model. This document describes the functional and non-functional requirements of the software, as well as the assumptions and constraints that are to be considered during its development. It aims to give an extensive and detailed guide for the development of software that can precisely estimate the depth of an object using a single picture or video frame, while adhering to a set of requirements.

1.2 Product Scope

The model being developed is to be implemented as an embedded software. The goal of this project is to:

a. create a highly effective, dependable, and efficient system that can precisely estimate the depth of objects in real-time while working within the limitations of the embedded system.
b. build a model should be capable of handling a variety of input situations and data types.
c. ensure that the software is optimised for performance, accuracy, and efficiency.

The final embedded software should be able to display a live feed from the vehicle's camera and notify the driver of an obstacle's proximity to their vehicle.

1.3 Intended Audience and Document Overview

The audience for this document encompasses both the group of professors as well as the team of developers and writers responsible for creating the product. The document itself provides a thorough overview of the functional and non-functional requirements of the product, as well as its functions, scope, and implementation. It outlines all the characteristics and capabilities that must be integrated into the final product. The entire document is organized into four sections:

1. Introduction
2. Overall Description
3. Requirements
4. Appendix

1.4 Definitions, Acronyms and Abbreviations

1. SRS – Software Requirements Specification
2. MDE – Monocular Depth Estimation
3. CPU – Central Processing Unit
4. GPU – Graphics Processing Unit
5. RGB-D – A combination of an RGB image and its corresponding depth map (image containing depth information)
6. TBD – To Be Decided
7. ADAS – Advanced Driver Assistance Systems
8. RADAR - Radio Detection and Ranging
9. LiDAR - Light Detection and Ranging

1.5 Document Conventions

Font conventions:

a. The font used is 'Times New Roman'.
b. All headings and subheadings are in bold.
c. The header text is in italics.
d. The sizes of the font used are as follows:
    i. Page Headings – 24 pt
    ii. Main Headings – 24 pt
    iii. Subheadings – 18 pt
    iv. Regular Text – 12 pt
    v. Header and Page Number – 11 pt

Spacing and alignment conventions:

a. There exists a 1 line gap between each main heading, subheading and paragraph.
b. The text is justified to the fit the pages.
c. Each inner set of numbered bullets are further indented than the outer set.
d. The text is left aligned, except for the title on the cover page, and the page numbers.

Figures and table conventions:

a. Every table is suitably numbered and with proper headings and column names.
b. The figures are also numbered correctly, with concise and accurate title.

1.6 References and Acknowledgements

[1] Masoumian, A., Rashwan, H. A., Cristiano, J., Asif, M. S., & Puig, D. (2022). Monocular depth estimation using Deep Learning: A Review. *Sensors*, *22*(14), 5353. https://doi.org/10.3390/s22145353

[2] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for Zero-shot cross-dataset transfer. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 44(3), 1623–1637. https://doi.org/10.1109/tpami.2020.3019967

[3] Solawetz, J. (2020, September 29). How to train a custom object detection model with YoloV5. Medium. Retrieved March 14, 2023, from https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-yolo-v5-917e9ce13208

[4] Azevedo, P. (2022, July 13). *BDD100K to yolov5 tutorial*. Medium. Retrieved March 14, 2023, from https://medium.com/@pedroazevedo6/bdd100k-to-yolov5-tutorial-213e4a67d54b

[5] *FiftyOne user guide*. FiftyOne User Guide - FiftyOne 0.19.1 documentation. (n.d.). Retrieved March 14, 2023, from https://docs.voxel51.com/user_guide/index.html

2. Overall Description

2.1 Product Overview

The Monocular Depth Estimation model is a software based on deep learning that utilizes cutting-edge computer vision algorithms to estimate the depth of objects using a single RGB image and generates its corresponding depth image. The model is designed to mainly assist in autonomous driving as well as ADAS, where it accurately estimates the depth of obstacles in a scene, without requiring the use of additional sensors or hardware such as RADAR or LiDAR systems.

The model works by taking a single RGB image or video frame as input and applying an MDE algorithm to generate the corresponding depth map, using which the depth of obstacles can be determined. It also features pre-processing and post-processing techniques to improve the accuracy and quality of the depth estimation.

The MDE model is highly scalable and can process input images or videos of various resolutions and sizes, without any constraints on the number of objects in the scene. We do, however, impose an artificial limit on the number of objects detected to simplify the model.

Overall, the Monocular Depth Estimation model is an essential tool not just for the automotive industry, but any industry that requires accurate depth estimation from a single image or video. The model can be integrated into various applications and systems, providing reliable depth information to aid in decision making and analysis.

2.2 Product Functionality

The functionality of monocular depth estimation is primarily focused on accurately estimating the depth of objects using a single RGB image or video frame. This is achieved through the following features:

a. Input Image Handling: The product can handle input images and videos in various formats such as JPEG, PNG, MOV and MP4.

b. Object detection algorithm: YOLOV5 is used as the object detection model, where the weights are generated through custom training of said model using the BDD100K dataset. It contains 10 categories of ground truth box labels and many objects from natural scenes, thus improving accuracy of the model to be specifically used for outdoor image detection purposes.

c. Depth Estimation Algorithm: Supervised MDE model is built using a Convolutional Neural Network that is capable of learning depth information of scenes by approximating the ground truth. These neural networks can predict depth maps from a single image.

d. Pre-processing: The format requirements of YOLOv5 are not met by the BDD100K dataset. Hence, we change the file and image/video formats to be compatible with the model using open-source tools.

e. Post-processing: The product performs post-processing on the estimated depth map to improve its quality. These techniques include depth map smoothing and filtering.

f. Real-time Performance: The product is designed to detect objects and estimate the depth of objects accurately in real-time.

g. Accuracy: The mean precision of detection can be improved by 1.9% in a custom trained YOLOv5 model as compared to a pre trained YOLOv5 model.

h. Scalability: The product can process input images of various resolutions and sizes, without any constraints on the number of objects in the scene.

2.3 Design and Implementation Constraints

A project of such nature involves a large number of constraints, both in terms of implementation, as well as the design of the product. These are:

a. GPU: A dedicated GPU with atleast 8 GB of VRAM is required for training large neural networks. For inference, a smaller GPU with around 2GB of VRAM should be sufficient.

b. CPU: A multi-core CPU with a base clock speed of atleast 2 GHz is required.

c. Memory: The system requires ample amount of RAM to train the neural network.

d. Storage: Deep learning models, as well as the accompanying datasets occupy a lot of storage. Hence, a bulk storage device such as a hard drive, or cloud storage options are required. The size of the datasets also leads to issues in uploading and downloading the data, which can be extremely time consuming.

e. Dataset availability : Datasets of compatible formats are unavailable, thus generating overheads in pre-processing of data.

f. Programming Language & Libraries: The deep learning model is entirely coded in Python using the PyTorch framework, and the data is pre-processed using the FiftyOne library.

3. External Interface Requirements

3.1 User Interface

Since it will be implemented as an embedded system, the user interface for the MDE model should be designed in such a way that the user can interact with the software in a simple and intuitive way while driving. The user should be able to control the software without being distracted from the road, and the UI should be compatible with the touchscreen and/or buttons present in the car's display system.

The driver can be alerted of the various obstacles using auditory methods, such as beeping sounds. This ensures that the driver is not required to take their eyes off the road to understand information from the software.

3.2 Hardware Interfaces

1. Processing: MDE models require significant processing power to produce accurate depth maps in real-time. Therefore, the hardware interface should be capable of supporting efficient processing. This includes components such as a powerful CPU and GPU.

2. Memory: The model requires a significant amount of memory to store the input data, intermediate results, and output data. The hardware interface for training should have a large amount of memory to support the model's requirements. However, once the model is trained, we do not require that the dataset be stored on the target device. Hence, only a small amount of storage is required on the target device.

3. Camera: The hardware interface should support a high-quality camera that can capture images/video with sufficient resolution and clarity, in the supported data formats. The camera should also ensure accurate and consistent capture, with little to no smearing or

motion blur. It should also be visible enough during low-light conditions, such as at night or during fog.

4. Supported Devices: While the end product is to run on an embedded system, the model can run on most modern architectures, such as 32 and 64-bit desktops. They can also be run on 64-bit Android phones, and 64-bit iPhones using packaging tools that allow the model to be distributed to those platforms.

3.3 Software Interfaces

1. Supported Operating Systems: The model can be run on most operating systems, which includes Windows, Linux, Android, iOS and macOS.

2. Programming Language & Libraries: We code the entire program in Python, using the following libraries:
   a. PyTorch
   b. NumPy
   c. Pandas
   d. scikit-learn

3. Development Environment: The model is developed in a Python environment, such as Jupyter notebooks or Google Colab. The latter is preferred due to availability of free cloud computing resources and collaboration features, enabling the developers to work together.

4. Other Non-Functional Requirements

4.1 Performance Requirements

The system should generate depth maps and infer the depth data either in real-time or within a reasonable time frame. The software should not be highly resource-intensive and should not impact the device's performance significantly.

4.2 Safety and Security Requirements

The system should be secure, protecting the internal data and preventing unauthorized access or tampering.

4.3 Software Quality Attributes

The software should have the following attributes:

1. Adaptability: The model must be able to adapt to a wide variety of changing conditions.

2. Correctness: Thorough testing is carried out to ensure the correctness of the model.

3. Flexibility: The model should work with different types of inputs and data formats.
4. Maintainability: Bugs in the program should be easy to repair and maintain.

5. Reliability: The product should perform reliably in different working conditions.

5. Appendix – Glossary

1. SRS – Software Requirements Specification
2. MDE – Monocular Depth Estimation
3. RGB-D – A combination of an RGB image and its corresponding depth map (image containing depth information)
4. ADAS – Advanced Driver Assistance Systems
5. RADAR - Radio Detection and Ranging
6. LiDAR - Light Detection and Ranging