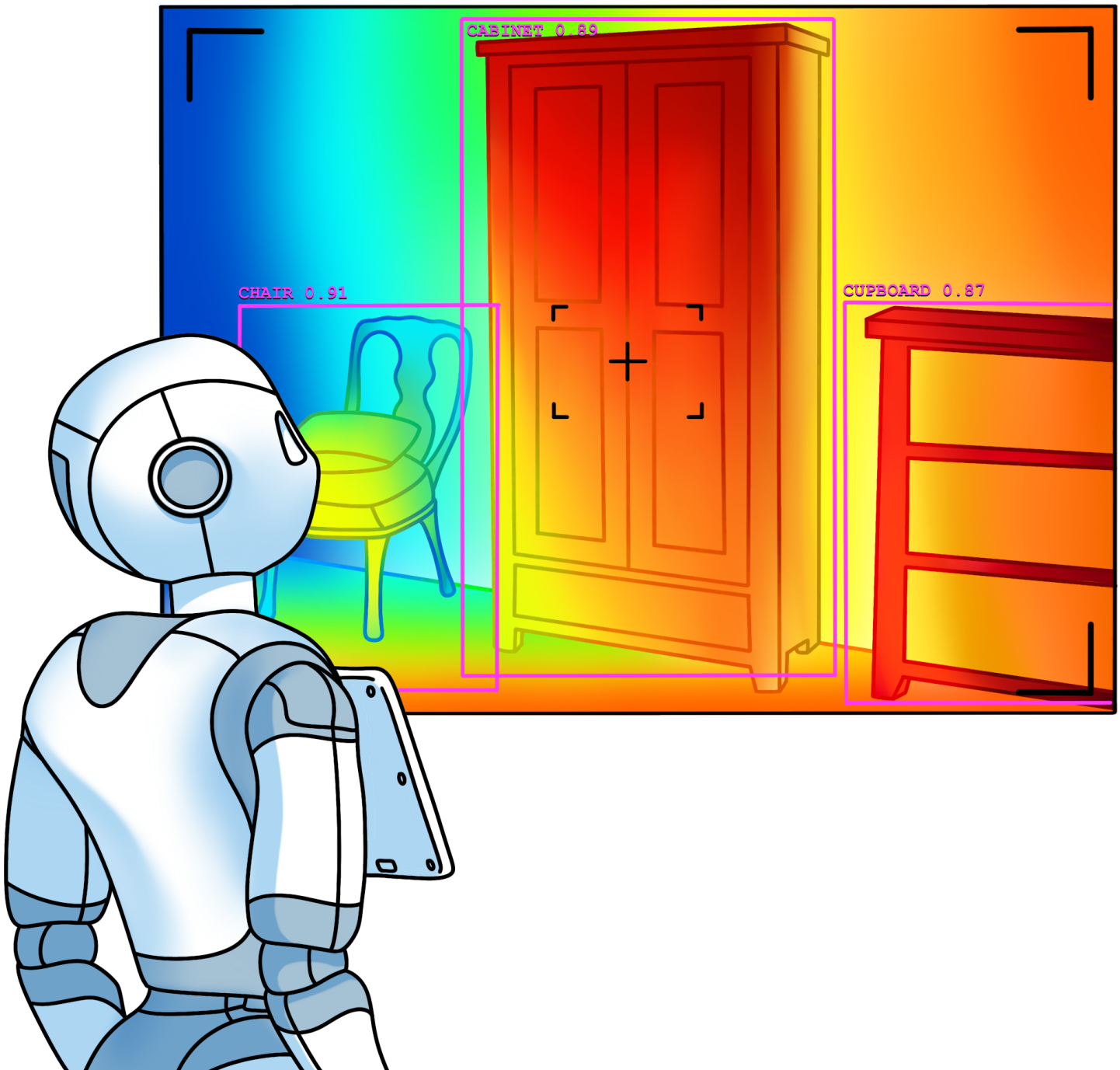
DTU Civil Engineering
Department of Applied Mathematics and Computer Science

Monocular Depth Estimation in Robotics

From Survey to Perception–Action with the Pepper Robot

Bachelor Project



Monocular Depth Estimation in Robotics
From Survey to Perception–Action with the Pepper Robot

Bachelor Project
January 3, 2026

By
Nicklas Mundt

Cover illustration: Kristine Fredslund Jacobsen, 2025

Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby Denmark

https://www.compute.dtu.dk/

Acknowledgements

I would like to thank my supervisor, Thomas Bolander, for his guidance, constructive feedback, and support throughout the work on this project.

Approval

This Bachelor of Science project was carried out over a period of 18 weeks at the Technical University of Denmark (DTU) in partial fulfillment of the requirements for the degree Bachelor of Science in Artificial Intelligence and Data.

The reader is assumed to have a basic knowledge of deep learning and computer vision.

Abstract

Monocular depth estimation (MDE) has emerged as an increasingly relevant technique in vision-based robotics, offering accessible depth perception without the need for dedicated sensing hardware. This project investigates the development of monocular depth estimation through a survey of its historical evolution, highlighting major paradigm shifts from traditional computer vision methods to modern deep learning and transformer-based approaches. Based on insights from the survey, a state-of-the-art monocular depth estimation model is selected and integrated into a perception–action system on the Pepper humanoid robot. The implemented system combines object detection and monocular depth estimation to enable the robot to localize, approach, and prepare for interaction with objects in a controlled environment. Experimental results demonstrate that modern monocular depth estimation is sufficiently robust to support coarse navigation and object approach behavior, despite limitations in absolute depth accuracy and sensing latency. These findings suggest that monocular vision provides an accessible and cost-effective alternative to depth sensors such as LiDAR or stereo cameras for certain robotic tasks. Overall, the project illustrates the practical potential of monocular depth estimation for enabling perception-driven robotic behavior, as the technology continues to advance.

The perception–action system as well as the experiments can be found in the *Github repository* on https://github.com/Feen-X/PepperMDE

Contents

1 Introduction

1.1 Background and motivation

Humanoid robots are no longer just a concept from science fiction but are increasingly becoming part of real-world applications. Boston Dynamics, Tesla, and Unitree Robotics are some of the many companies leading the race to develop humanoids that can operate in real-world environments and be integrated into different industries. In recent years, the field has gained massive attention and investment, as progress in robotics and artificial intelligence has made these machines increasingly capable and versatile. According to a 2024 forecast by Goldman Sachs, the global market for humanoid robots is expected to reach 38 billion USD by 2035 [1], an increase of more than six times their earlier estimates.

What makes humanoid robots particularly interesting is their potential to operate in environments designed for humans. Unlike traditional industrial robots that excel at repetitive and well-defined tasks, humanoids are built to handle variation and uncertainty. While they may not outperform specialized robots in speed or precision, their strength lies in their ability to adapt and function in dynamic, human-centered settings [2].

To operate effectively in human settings, robots must be able to perceive and understand their surroundings, a task that is natural for us humans but remains a central challenge for machines. In particular, perception pipelines need to recognize and localize objects to enable meaningful interaction. Object localization is typically achieved through depth estimation, commonly using specialized sensors such as LiDAR or stereo cameras, or a combination of multiple. While these sensors provide accurate depth information, they are often expensive, only adding to the overall complexity and expense already associated with robotic systems.

Computer Vision based methods offer an attractive alternative, as they infer spatial information directly from visual input in a way that resembles human perception. Among these, **monocular depth estimation** has gained attention for its ability to extract depth from a single RGB image, thus also being a cheaper alternative. And with the introduction of deep neural networks, the field has really fostered the last 10 years [3], with some of the first deep learning methods emerging around 2014 [4].

In this bachelor project, the evolution of monocular depth estimation is explored through a survey, showcasing key paradigm shifts and influential contributions that have shaped the field. Based on insights from this survey, a modern monocular depth estimation model is selected and applied in a practical setting, where it is integrated into a perception–action system on the Pepper humanoid robot. By combining object detection with monocular depth estimation, the project investigates whether monocular vision alone can reliably do object localization and approach behavior in a controlled environment, without relying on dedicated depth sensing hardware.

1.2 Report Structure

This report is structured as follows. Chapter 2 presents the problem definition and analysis. Chapter 3 then outlines the chosen methodologies, introducing the overall system concept from a technical perspective and the key techniques used throughout the project. Chapter 4 presents a survey of monocular depth estimation, tracing its historical development and recent advances. Chapter 5 describes the design and implementation of the

perception–action system on the Pepper robot, and covers the experimental setup. Chapter 6 presents the evaluation results and discusses the limitations of the system. Finally, Chapter 7 concludes the report with a summary and directions for future work.

2 Problem Definition and Analysis

In this chapter, a more in-depth description of the problem will be provided and analyzed, and the constraints and assumptions of this project will be discussed.

2.1 Problem Context and Motivation

A crucial part of any autonomous system or robot lies in its ability to perceive and interpret its surroundings. For a robot to interact with the world meaningfully, it must not only recognize relevant objects, but also understand their relationships within a broader context. One typical such relationship could be *where* the objects are and *how* they can be reached. This coupling between perception and action forms the foundation of this project.

The motivation for this work originates from scenarios in which a robot is expected to assist a human in achieving a certain goal. Situations like this illustrate the importance of a perception–action pipeline, where the robot first has to detect relevant objects in the environment, and then make an action accordingly. This concept serves as the basis for the project, in which the experimental evaluation will focus specifically on enabling the Pepper robot to identify objects of interest and approach them using vision-based perception.

However, achieving this requires a robust and reliable estimate of the distances to objects, a task that remains a challenge for robots today. Traditional methods such as LiDAR or stereo vision demand additional sensors or calibration, increasing the complexity and cost of the system. Monocular depth estimation, on the other hand, provides a promising alternative, as it allows depth perception using only a single RGB camera, utilizing deep learning. Combined with deep learning approaches for object detection as well, it opens the possibility for accessible, vision-based perception systems capable of supporting autonomous goal-directed interaction.

2.2 Problem Definition

Building upon the motivation stated above, this project addresses the challenge of enabling a robot to perceive and act upon its surroundings according to a goal, using only monocular visual input. Specifically, it investigates whether modern monocular vision-based methods can provide sufficiently reliable spatial information to allow the Pepper robot to detect relevant objects in its environment, estimate their distance, and perform goal-directed actions based on that understanding.

Since depth perception is a fundamental part of this perception–action pipeline, the project places particular emphasis on the primary source of distance information: monocular depth estimation. Understanding how it has evolved and how it can be applied in practical robotics provides both the theoretical and technical basis for this work.

From this, the overall problem can be formulated as follows:

Problem Formulation

What can be learned from monocular depth estimation, and how can such vision-based methods be applied to enable the Pepper robot to perceive and act upon objects in a controlled environment?

To address this overarching problem, the work is divided into two main components, each with its own set of objectives.

The first is an **analytical component**, presented in the form of a *survey*, which aims to provide a foundation for the practical work by:

1. Investigating the historical evolution of monocular depth estimation and its major paradigm shifts.
2. Analyzing recent methods and identifying state-of-the-art approaches suitable for robotic applications.

The second is a **practical component** that focuses on implementing a *perception–action system* on the Pepper robot. This involves the following sub-goals:

1. Object detection and identification: Detect and recognize a specific object in the visual scene.
2. Depth estimation: Estimate the distance between the robot and the object using a selected monocular model.
3. Motion control: Use the combined perception output to guide the robot's movement toward the target and stop at an appropriate distance.
4. Interaction: Enable interaction readiness rather than full physical manipulation.

Together, these components and sub-goals define the technical scope of the project and provide a foundation for the following analysis and design stages.

2.3 Constraints and Assumptions

To keep the project measurable and within a realistic scope, certain constraints and assumptions have been defined. These serve to limit the complexity of the system while allowing a focused evaluation of the core objectives.

2.3.1 Hardware Constraints

The Pepper robot is equipped with two identical RGB cameras located in the forehead, each capable of providing an image resolution of up to 2560×1920 at 1 frames per second (fps) or 640×480 at 30 fps, according to its documentation [5]. In practice, however, the frame rate varies between individual Pepper robots.

These limitations in fps and image acquisition introduce significant constraints on the responsiveness of especially real-time vision-based perception pipelines. But while the system architecture allows for the use of an external camera as the primary visual input, this option was not employed in the present implementation. The impact of choosing the integrated camera, as well as the potential benefits of using an external camera, are discussed further in Chapter 6.

Due to Pepper's limited computational power, all deep learning models are executed on an external computer. The robot connects to this computer via a local Wi-Fi network, where a local server handles inference and control communication.

2.3.2 Environmental Constraints

The experiments with the system are conducted in an indoor environment with stable lighting conditions, limited space, and minimal background clutter. The goal is to ensure consistent perception and reproducible results while demonstrating the concept under controlled conditions. Outdoor or highly dynamic environments are therefore considered out of scope.

2.3.3 Project Scope and Assumptions

The focus of this work is on visual localization; enabling Pepper to detect and estimate the position of relevant objects using monocular depth estimation. While object detection plays a crucial role in the perception pipeline, a detailed analysis of detection methods is not the primary focus of this project. Instead, one robust detection approach is selected from a pilot study and used consistently throughout the remaining work to ensure a stable perception foundation. The choice procedure is discussed in the next chapter, in Section 3.2.

Regarding the objects of interest to be detected in the experiment, they are assumed to be known in advance or selected from a limited, predefined set, and to represent static structures; more information on that in Chapter 5.4.

The monocular depth estimation model used is assumed to be pre-trained on suitable data (e.g., indoor scenes) rather than trained from scratch, due to computational and time constraints. As for the performance, real-time is desired but not strictly required. Minor inference delays are acceptable as long as the overall perception–action loop remains functional and coherent.

3 Selected Methods

As mentioned, enabling the Pepper robot to perceive and act upon objects is no simple task to solve. There are many different approaches one can take to achieve the same result. In this section, the chosen approach for this project will be stated, and a brief introduction to the methods and technologies used will be given. Also, before going in-depth with monocular depth estimation, a general overview of the task of estimating depth will be covered.

3.1 System Concept and Approach

A perception–action system on the Pepper robot can be designed in several ways, depending on multiple factors such as the sensing capabilities or the computational constraints. In this project, the overall approach is structured around a sequential pipeline consisting of three core stages: object recognition, depth estimation, and action execution. Separating these components makes the system more modular and adaptable, while ensuring that each stage delivers the information required by the next.

The first stage focuses on object recognition. Here, a lightweight yet reliable detector is employed to identify the target object in the camera feed from the robot. The choice of detection model is discussed in the following section, where different alternatives are evaluated before selecting the one used in this system.

Once an object has been detected, its approximate distance from the robot must be inferred in order for Pepper to interact with it meaningfully. Depth estimation methods vary widely in their hardware requirements, accuracy, and computational cost. Because of this diversity, and because depth plays a central role in the system's overall performance, the most relevant depth estimation approaches are introduced later in this chapter. As will become clear, the project ultimately focuses on monocular depth estimation, which is examined in greater depth through a dedicated survey in the following chapter.

Together, these two perception components supply the information necessary for the action module, which uses the estimated object position to guide Pepper's behavior. This modular perception-action framework forms the basis of the localization and interaction system developed in the remainder of this report. For the more in-depth description of its implementation, see Chapter 5.

3.2 Object detection with YOLO

Object detection is a deeply researched area in computer vision, as it has an abundance of practical use-cases in many fields. In the early days of object detection (around the 2000's) manual feature extractors were used to detect objects, such as facial features [6]. Although some of the earlier methods are still used by large companies today [7], they have also evolved into more robust and reliable alternatives by using neural networks and deep learning. In this branch of object detection, multiple new approaches have been introduced.

The YOLO (You Only Look Once) algorithm is a prominent example of early single-shot object detection approaches. Introduced in 2016, YOLO reframed object detection as a single regression problem, requiring only a single forward pass through a convolutional neural network to predict bounding boxes and class probabilities [8]. Concretely, the input image is divided into a grid, where each grid cell is responsible for predicting object

locations and corresponding class labels within its region. This single-pass design makes YOLO highly computationally efficient and well suited for real-time applications, although it can struggle in scenes containing densely packed objects [9].

Since its introduction, YOLO has remained competitive through continuous architectural refinements across multiple versions. Modern implementations, such as those developed by Ultralytics, extend the original framework beyond object detection to related computer vision tasks including image segmentation and pose estimation [10].

In this project, YOLO is an appropriate choice to use due to its proven balance of robustness, speed, and low computational demands. The lightweight nature of the model is especially important here, as additional deep learning components will be introduced later in the pipeline. Ensuring that detection can run efficiently in real time therefore requires avoiding unnecessarily heavy architectures.

Three different detector models were initially considered. First, YOLOv8, a widely adopted model known for its strong accuracy and reliability [11]. The second was YOLO World, a more recent framework built on the YOLOv8 backbone and designed for open-vocabulary object detection [12]. Finally, the most recent model in the YOLO family, YOLOv11 [13], was proposed for its substantial improvements in speed and performance.

At the beginning of the project, YOLO World appeared to be the best fit, as open-vocabulary detection offers considerable flexibility. Unlike conventional models constrained by a predefined set of labels, open-vocabulary detectors, popularized by models such as Google's OWL-ViT [14], can identify previously unseen object categories using a vision-language model. These systems typically pair a detector with a model such as CLIP [15], allowing objects to be recognized based on their similarity to text embeddings. This capability seemed advantageous for the project, since the target objects (e.g. ”a white cabinet” or ”a small handle”) could be defined simply through text prompts.

However, practical experimentation revealed several limitations. While YOLO World reliably detected larger, more general objects such as a cabinet, it struggled to identify smaller or more fine-grained elements, such as the handle on the cabinet. It also exhibited difficulty in distinguishing objects based on specific attributes such as color, for example differentiating between a red chair and a blue chair. These inconsistencies made it unsuitable for the level of precision required in the project.

Given these challenges, the choice of the model was revised. YOLOv8 was briefly adopted, but after a short pilot study, YOLOv11 was selected instead. YOLOv11 demonstrated noticeably greater robustness and accuracy compared to its predecessor while maintaining high inference speed, making it the most reliable option for the real-time detection requirements of the project.

3.3 Localization by Depth Estimation

Estimating depth in an environment is a fundamental requirement for 3D object detection and, more broadly, for any real-world robotic system and application. Depth estimation enables a robot to infer the three-dimensional structure of its surroundings, typically represented as a depth map. This spatial understanding is essential for tasks such as navigation, manipulation, obstacle avoidance, and scene perception. The importance of depth information is reflected in the wide range of methods that have been developed to obtain it, each with different strengths, limitations, and application scenarios. This section provides an overview of some of the most widely used approaches to depth estimation.

3.3.1 LiDAR methods

One of the most prominent methods of depth estimation today relies on LiDAR technology. LiDAR, short for *Light Detection and Ranging*, measures distance by emitting rapid pulses of laser light and recording the time it takes for each of the pulses to return after reflecting from objects in the environment. The resulting measurements are collected as a point cloud, which can then be converted into a highly accurate 3D representation of the surroundings.

Despite its precision and reliability, LiDAR is not without limitations. Its performance has been shown to degrade under challenging weather conditions, as rain, fog, and other reflective surfaces can interfere with the laser signal. To mitigate these drawbacks, LiDAR is commonly coupled with complementary sensors such as cameras, radars, or ultrasonic sensors [16].

The most significant drawback of LiDAR for this project, however, is its cost. LiDAR sensors remain substantially more expensive than camera-based alternatives, limiting their accessibility and suitability for budget-constrained systems. For this reason, among others, LiDAR-based depth estimation was not selected for use in this project.

3.3.2 RGB-D methods

The RGB-D methods combine normal images with depth information. The RGB component can be obtained using a standard camera, while the depth component can be produced using a variety of sensing techniques [17]. Two commonly used approaches for acquiring depth in RGB-D systems are:

- **Time-of-Flight (ToF) cameras:** These sensors estimate depth by measuring the time it takes for emitted infrared light to travel to an object and back, similar in principle to LiDAR. In practice, they are significantly more cost-effective and less complex than LiDAR systems. However, ToF cameras generally operate at lower spatial resolutions and over shorter ranges [18].
- **Stereo vision:** Stereo methods use two spatially separated cameras to infer depth geometrically. By comparing disparities between the two captured images, the system can estimate the distance to objects. Stereo vision is also notably cheaper than LiDAR, but it does have certain disadvantages, such as being limited by its relatively short effective range that is dependent on image resolution as well as the need to have adequate lighting conditions. Poor illumination or texture-less surfaces can also lead to degraded performance [19].

Although RGB-D methods offer a more affordable alternative to LiDAR and are more feasible to integrate into platforms such as the Pepper robot, they still come with notable limitations. Furthermore, even these approaches introduce additional hardware costs. Ideally, if depth estimation only utilized a standard monocular camera, it would become even cheaper. This is explored in the next section.

3.3.3 Monocular vision

Monocular vision approaches estimate depth using only a single RGB camera. In contrast to LiDAR and RGB-D systems, which require specialized and often costly hardware, monocular methods rely solely on image information that is easy to integrate on most robotic platforms. This makes them lightweight, inexpensive, and attractive for systems with strict hardware and computational constraints.

For this project, monocular vision offers several key advantages over LiDAR and RGB-D alternatives as further illustrated in Table 3.1. It avoids the financial and mechanical

Estimation Method		Efficiency	Complexity	Robustness	Price
LiDAR		High	High	Very High	Very High
RGB-D	ToF	Medium	Medium	High	Medium
	Stereo	Medium	Medium	Medium	Low
Monocular		High	Low	Medium	Very Low

Table 3.1: Simple comparison of common depth estimation methods. Monocular depth estimation is lightweight and cost-effective, making it suitable for this project.

overhead of additional sensors, improves on the range limitations of stereo or ToF-based systems, and integrates seamlessly with the camera hardware of the Pepper robot. Moreover, because depth estimation models can be executed on an external computer, the approach circumvents the onboard processing limitations of the robot while still allowing real-time performance.

While several monocular approaches rely on external cues – such as fiducial markers, or Structure-from-Motion pipelines – to infer depth, these methods often depend on strong assumptions or practical setup. To reduce these dependencies, the project focuses exclusively on methods that infer depth directly from a single image, without requiring any additional sensors, motion constraints, or engineered visual aids.

Given the advantages of monocular depth estimation, it stands out as the most suitable method for this project. The next chapter therefore provides a survey that dives deeper into monocular depth estimation techniques, tracing their development, and analyzing the most influential models. This survey forms the foundation for selecting and implementing the models used in the perception–action object localization system in this project.

4 Monocular Depth Estimation - A Survey

Monocular Depth Estimation (MDE) is the computer vision act of estimating the per-pixel depth distances through a single RGB image. Mathematically, it can be formulated as: from an image $I \in \mathbb{R}^{w \times h}$ with dimensions $w \times h$, the goal is to formulate a non-linear per-pixel mapping $\Psi : I \rightarrow D$ into a depth map D with the same dimensions as I. However, the act of inferring depth from a single image poses a fundamental challenge:

> "*Given an image, an infinite number of possible world scenes may have produced it.*" – Eigen et al., One of the pioneers in deep learning MDE

This statement highlights the ill-posed nature of monocular depth estimation; scale ambiguity and missing spatial information make depth inference inherently uncertain – a challenge that has persisted since the very beginning of the field.

To understand how people are tackling MDE today, the history of the field is covered in this chapter. An overview of the topics and milestones discussed can be seen in Figure 4.1. The chapter is organized accordingly: section 4.1 discusses the transition from geometric to learning-based approaches, section 4.2 reviews the evolution of neural architectures, section 4.3 examines different learning paradigms, section 4.4 highlights changes in how depth itself is represented, section 4.5 concludes with the latest frontiers and open challenges, and ends with a summary in section 4.6.

4.1 From Geometry to Learning

The traditional methodologies for MDE can be traced back to the early 2000s. Before the rise of machine learning, the traditional methods relied primarily on geometric cues and hand-crafted features to infer depth from single images. Classic approaches included perspective geometry [20], which exploited the observation that parallel lines appear to converge toward a vanishing point in the distance – like train tracks going towards the horizon – and atmospheric scattering, where objects far away appear less saturated and slightly bluish due to light diffusion in the air.

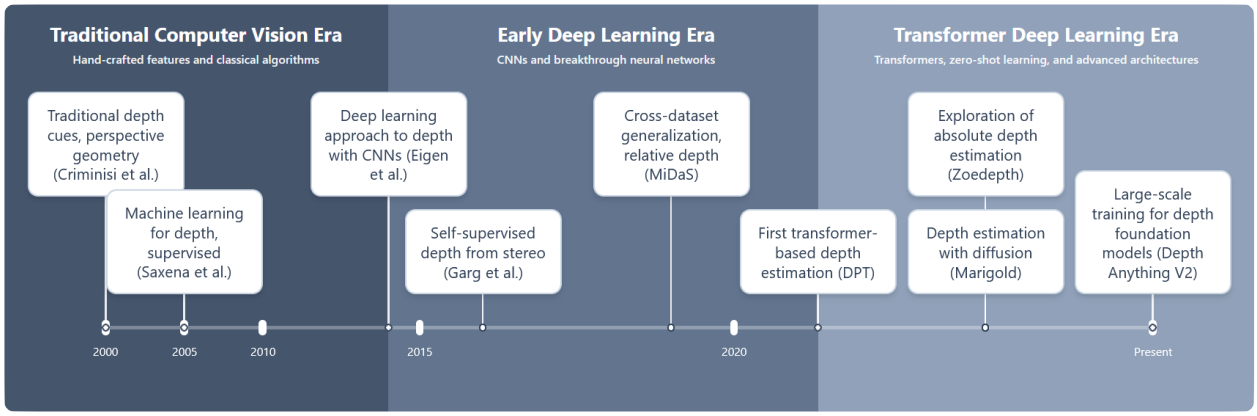


Figure 4.1: A simple timeline over relevant breakthroughs in the field of Monocular Depth Estimation. In this project, it is split up into three phases, to illustrate paradigm shifts.

Although these methods to some extent provided a solution to the inherent scale ambiguity problem, they lacked robustness and generalization. They often relied on strong assumptions, such as consistent lighting, uniform textures, or planar surfaces. With the emergence of machine learning and later deep learning, it became evident that data-driven models could overcome many of these limitations by learning depth cues directly from visual data, leading to a major leap in both accuracy and adaptability.

One of the first to apply machine learning techniques was Saxena et al. [21], who approached the problem using supervised learning. They introduced a discriminatively trained Markov Random Field (MRF) model that incorporated multi-scale local and global image features to predict depth in unstructured outdoor scenes. This work marked a turning point toward machine learning-based monocular depth estimation, inspiring several follow-up studies that further explored MRFs for this task [22, 23].

4.2 Evolution of Model Architectures

As the field of computer vision has evolved, so has the field of MDE. In the following, the evolution of the architecture of MDE has been categorized into 3 distinct approaches.

4.2.1 Early CNN Approaches

A major leap in the MDE scene came in 2014 with Eigen et al. [4], who introduced the first deep learning neural network for MDE. Their work represented a major departure from previous hand-crafted or probabilistic methods by allowing the model to learn depth cues directly from large-scale image data using supervised methods. Their two-stage *Convolutional Neural Network (CNN)* architecture captured both coarse global structures and fine local details.

The problem with the scale ambiguity is that the estimated depth D of the ground-truth depth D^* might be geometrically correct, but differ with a slight global scaling factor $c > 0$:

$$D^* = c\,D.$$

A standard loss function like $\sum_i (D_i - D_i^*)^2$ would heavily penalize this offset, even if the relative depth relationships are completely correct. Eigen et al. addressed this inherent challenge through the introduction of a *scale-invariant loss function*, explicitly designed to account for the scale ambiguity present in monocular depth estimation. Their loss function

$$E(D, D^*) = \frac{1}{2N} \sum_{i=1}^{N} (log\, D_i - log\, D^*{}_i + \alpha(D, D^*))^2$$

ensures that predictions which are incorrect only by a constant global scale only gives a small penalty. This is achieved by modifying the standard loss function by adding the term $\alpha(D, D^*) = \frac{1}{N} \sum_i (\log D_i^* - \log D_i)$, which finds the optimal offset that best aligns the predicted depths with the ground truth. Additionally, operating in log-depth space transforms multiplicative scale errors into additive offsets, while normalization by N makes the loss independent of image resolution.

As a result, the loss ignores errors that can be fixed by scaling and only penalizes errors in relative depth, rather than absolute depth values. Equivalently, the loss can be interpreted as the variance of the log-depth errors, encouraging the network to learn coherent scene geometry and correct depth relationships rather than absolute depth values.

With this formulation, Eigen et al. achieved state-of-the-art performance on both the NYU Depth v2 and KITTI benchmarks at the time. In subsequent work, they further improved their approach through deeper architectures, higher-resolution predictions, and refinements to the loss function, leading to more accurate and detailed depth estimates [24].

4.2.2 Deep Residual and Encoder-Decoder Designs

In the following years of the publication by Eigen et al., numerous studies were made to improve the CNN architecture and training process. A key contribution came from Laina et al., which introduced a fully **convolutional residual network** for depth estimation [25]. This architecture improved both accuracy and efficiency. Using residual learning and encoder-decoder structures, their model achieved a much deeper architecture with fewer parameters, making it much faster and giving a better generalization. This work effectively popularized the use of residual and encoder–decoder structures within depth estimation.

Building upon these ideas, further developments such as DORN (Depth Estimation via Ordinal Regression) [26] reframed the regression problem into an ordinal classification task, allowing the network to learn relative depth orderings rather than raw regression values, improving stability and edge preservation. These innovations marked the maturity of supervised CNN-based depth estimation, again achieving impressive accuracy on datasets such as KITTI and NYU Depth v2.

4.2.3 Transformer-based Architectures

Despite their success, convolutional models inherently suffered from limitations in capturing global context, as individual convolutional operations are restricted to local receptive fields. For them to model larger contexts, CNNs have to rely on stacking layers and doing repeated downsampling, making the models very deep. Although various architectural improvements have been proposed to mitigate these effects as discussed in the previous section, global context modeling in purely convolutional frameworks remains challenging, particularly for MDE that needs to be able to sometimes capture very fine details in the image.

The introduction of the transformer architecture [27] offered a compelling alternative through its popular attention mechanisms. With the publication of *Vision Transformers (ViT)* [28], this idea of attention was successfully adapted to visual data, sparking a major shift in computer vision research and marking the beginning of the transformer era for vision tasks.

This shift also influenced the field of MDE, with the first ViT-based approaches being led by the introduction of *Dense Prediction Transformers (DPT)* [29]. Unlike CNNs, the transformer backbone employed in DPT performs a single initial image embedding, splitting it into tokens, and then maintains a representation with the same dimensionality throughout all processing stages, avoiding repeated spatial downsampling. Through self-attention, each token directly incorporates information from all other tokens, enabling global context to be modeled at every layer without sacrificing spatial detail. This design is particularly well-suited for monocular depth estimation, where accurate depth prediction requires both global scene understanding and precise spatial correspondence.

The success of the DPT architecture established transformers as a new general framework for MDE, marking a new era for MDE, as illustrated in Figure 4.1. Today, most work is focused around this framework, some of which will be covered later in this chapter.

4.3 Evolution of Learning Paradigms

Like the architecture of the models, the *different ways* of training them have also been explored throughout the deep learning era.

4.3.1 Supervised Learning

Early deep learning methods such as Eigen et al. and Laina et al. were supervised, using an image and its corresponding ground-truth depth map to train the model. These

ground-truth depth maps are typically obtained from LiDAR, structured light sensors, or stereo reconstruction. While the method itself is accurate, the annotations are costly and sensor-dependent, and can be noisy or incomplete, thus restricting generalization.

4.3.2 Self-Supervised and Unsupervised Learning

To reduce this dependence on labels from supervised learning, researchers began exploring what were initially termed *unsupervised* approaches, but later described as *self-supervised*, since they rely on supervisory signals derived from the data itself.

One of the earliest examples is the work of Garg et al. (2016) [30], which trained a monocular depth network using stereo image pairs. Instead of learning from ground-truth depth, the model predicted depth from the left image and used the resulting disparity to warp the right image into the left view. Training then minimized a loss between the reconstructed left image and the original left image. In this way, the stereo pair replaced the need for explicit depth supervision. This idea was extended by Godard et al. with MonoDepth [31], which introduced a left–right consistency loss that substantially improved stability and accuracy. These works demonstrated that high-quality depth estimation is possible without any explicit depth supervision.

A related line of work explored *semi-supervised* learning, where a small amount of ground-truth depth is combined with self-supervised photometric losses. Kuznietsov et al. showed in 2017 that such hybrid approaches can outperform purely supervised or purely self-supervised methods [32].

Together, these methods made for a compelling alternative to traditional supervised training, offering broader scalability. However, they also introduced challenges, including higher training complexity and sensitivity to lighting and appearance changes [33]. As a result, supervised, semi-supervised, and self-supervised approaches all remain in active use today, with the choice of paradigm depending on the requirements and constraints of the specific application.

4.3.3 Zero-shot and Cross-Dataset Generalization

Despite these advances of the learning methods, they still lacked robustness outside their training domain. This sparked interest in zero-shot approaches aimed at strong cross-dataset performance.

Around 2019, Ranftl et al. introduced MiDaS (Mixed Dataset Depth Estimation) [34], which combined multiple diverse datasets with a scale- and shift-invariant loss in a CNN architecture. This enabled the network to produce cross-domain generalization, allowing the model to predict depth for unseen images without retraining. Although MiDaS was based on a CNN backbone, its emphasis on dataset-agnostic training and relative depth estimation established a new standard for real-world applicability, and directly inspired later transformer-based methods such as DPT. Furthermore, by leveraging on the transformer based architecture, MiDaS itself acquired substantial improvements, demonstrating even stronger cross-domain performance [35].

4.4 Evolution of depth representations

An important conceptual evolution in MDE lies in how depth itself is represented and learned. In the early days of MDE, the geometric methods were used to estimate absolute depth. However, this saw a change in the deep learning era of MDE.

4.4.1 From Absolute to Relative Depth

Early deep learning approaches to monocular depth estimation, such as those by Eigen et al. [4, 24], focused on predicting absolute depth values. However, these methods

revealed a fundamental limitation as earlier stated in this chapter: the inherent ambiguity of global scale in monocular vision. The proposed scale-invariant loss function from Eigen et al. reduced sensitivity to absolute scale differences by emphasizing the relative depth relationships between pixels, as mentioned in subsection 4.2.1. This idea highlighted the importance of capturing the geometric structure of a scene, rather than its absolute metric scale. Consequently, this subsequent research began to move away from purely absolute depth estimation and toward relative or scale-invariant formulations [3, 33]. This new approach of reformulating the task as Relative Depth Estimation (RDE) improved generalization across datasets. One key example is MiDaS, as already mentioned.

4.4.2 The Return to Absolute Depth Estimation

While RDE formulations improved generalization across datasets, they also prove insufficient in applications requiring accurate metric distances – such as robotics, autonomous driving, and AR – since RDE inherently trades metric precision for robust generalization. As a result, recent research has sought to reintroduce **monocular metric depth estimation** (MMDE) while maintaining the generalization benefits of scale-invariant training [36].

Some emerging solutions used camera parameters to help estimate absolute depth [37]. The more recent breakthrough of ZoeDepth, which extended the relative depth model MiDaS with techniques like adaptive metric binning and scene-aware routing, resulted in great generalization performance while outputting absolute scale [38]. Moreover, the recent wave of generative and multi-view approaches (e.g., diffusion-based models, NeRF-style 3D reconstruction) has further bridged the gap between relative and absolute depth [3, 33]. These methods leverage spatial and temporal consistency to recover an implicit notion of scale, enabling accurate metric predictions even without direct LiDAR supervision.

Overall, this return to metric depth reflects a maturation of the field: rather than abandoning absolute scale, contemporary models learn to infer it through richer contextual cues, bigger datasets for training, and 3D-aware generative priors. In doing so, they combine the robustness of relative representations with the practical utility of metric depth estimation.

This trend continues into the latest frontier of monocular depth estimation, where discriminative and generative models are increasingly unified. Generative methods, in particular, often recover metric depth as an emergent property of scene understanding, hinting at a convergence between depth estimation and general 3D perception.

4.5 The Latest Frontier

Despite the impressive progress of the deep learning era of MDE, several key challenges remain: achieving accurate metric depth in arbitrary scenes, producing high depth detail even on thin structures, generalizing to different visual domains (e.g., aerial, underwater, indoor vs outdoor seamlessly), and integrating depth estimation into broader 3D perception systems (e.g., robotics, mixed reality). In response, recent research is branching into two complementary directions, the *discriminative path* and the *generative* path.

4.5.1 Discriminative Foundation Models

The discriminative path builds on the large-scale training of depth networks, leveraging massive synthetic or pseudo-labeled datasets, scalable architectures in the form of transformers, and fine-tuning for metric accuracy.

A prime example is Depth Anything V2 (Yang et al., 2024) which trains a teacher-student

paradigm: a large transformer model is trained on synthetic images (the teacher). This model is then used to produce pseudo labels for tens of millions of real images, which the student model fine-tunes on. For this project, they both released relative depth models and metric depth models [39]. Depth Anything V2 demonstrates that by combining synthetic supervision, massive pseudo-label scale, and modern architecture, one can achieve unprecedented generalization and metric prediction capability. Such models are increasingly being considered “foundation models” for monocular depth, analogous to large language models in Natural Language Processing, with downstream fine-tuning for robotics, AR, and other domain specific tasks. Other examples of popular Vision foundation models include ViT, CLIP, and DINOv2 [28, 15, 40]. These models are adopted for a wide range of computer vision tasks including depth estimation.

4.5.2 Generative Methods

While discriminative methods focus on direct regression of depth from visual features, a parallel line of research has emerged that approaches depth estimation as a generative process.

One of the most influential works in this direction is Marigold [41], which repurposes a Stable Diffusion model for monocular depth estimation. Starting from a pretrained generative model, Marigold fine-tunes the diffusion model on large-scale synthetic datasets with paired depth information, allowing it to generate accurate depth maps without any explicit metric supervision from real-world data. Remarkably, it demonstrates strong zero-shot generalization across a wide range of scenes and domains, highlighting the robustness of generative priors in capturing geometric relationships.

The most recent version, Marigold 1.1 [42], further refines this concept by significantly improving inference efficiency – reducing the number of diffusion steps from 10–50 in the original model to as few as 1-4 – while simultaneously improving accuracy and consistency. This efficiency gain illustrates that generative depth estimation is rapidly becoming not only conceptually elegant but also practically viable for real-time or large-scale applications.

Beyond Marigold, several other works extend generative depth estimation beyond diffusion-based models. Some approaches continue to explore the diffusion paradigm [43], and others explore entirely different generative mechanisms. For example, Flow Matching techniques [44] replace the stochastic diffusion denoising process with deterministic flow dynamics, enabling faster and more stable depth generation while maintaining good fidelity.

This new generative approach represents a powerful shift in how depth estimation is conceptualized. Instead of learning a direct pixel-to-depth mapping, these models learn to reconstruct the underlying 3D structure of a scene as an internal latent representation and then infer the depth from that representation. In essence, depth becomes a natural product of comprehensive 3D scene understanding.

4.6 Summary

Over two decades, monocular depth estimation has evolved from handcrafted geometric rules to data-driven and transformer-based architectures capable of near-human depth perception. Yet, the fundamental ambiguity noted by Eigen et al. still remains, a solid reminder that perception, even for machines, is an act of inference under uncertainty.

5 System Design and Implementation

This chapter describes the design and implementation of the system developed for enabling the Pepper robot to detect a given object, estimate its distance using a monocular depth estimation model, and conceptually prove interactability. The design combines deep learning–based perception with a Finite-State Machine architecture for control and some multithreading for efficient real-time performance.

5.1 System Pipeline Overview

The system follows a perception–action pipeline that is managed by a Finite-State Machine (FSM). This architecture allows the robot to dynamically transition between behaviors depending on sensory input.

The perception techniques is the core component of the pipeline, as it handles the visual understanding required for decision making. As illustrated in Figure 5.1, each RGB image is processed by two deep learning modules: an object detection model and a monocular metric depth estimation model. Both uses the current camera frame as input. The object detection module identifies and localizes relevant objects within the image, providing bounding box coordinates for each detection, and focuses on the one with the highest confidence. The depth estimation module – that is the discriminative Depth Anything V2 model covered in Chapter 4.5 – combines this with the camera input to predict the metric distance to the area covered by the focused object. The combined output of these modules therefore provides both the class and estimated distance to the relevant object. This information is subsequently used by the FSM to determine the next appropriate state transition and action for the system.

The system is primarily being run in a sequential pipeline as shown in Figure 5.1, with the exception of a small parallel camera thread used to continuously buffer the latest video frames. This ensures that perception always works with the most recent input without delaying the main processing loop.

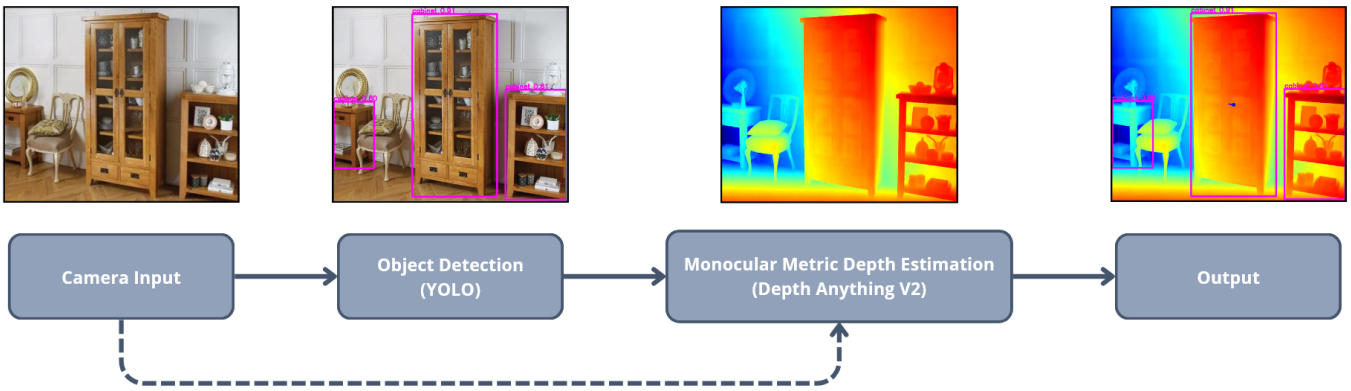


Figure 5.1: Illustration of the perception pipeline, focused on getting the depth of a given object. In the final output, the depth is estimated of the focused object, indicated with the blue dot.

5.2 System Architecture and Design Choices

Before diving into the detailed implementations, the architectural decisions will be covered to establish an understanding of the fundamental flow of the system.

5.2.1 Finite State Machine Structure

The system is built around a Finite-State Machine (FSM) structure, chosen for its modularity and clear interpretability when defining discrete robot behaviors. This type of structure excels in robotics tasks, as it allows the system to transition predictably between well-defined states based on sensory input and environmental conditions. This aligns naturally with the requirements of this project, where the robot's behavior can be effectively divided into distinct operational phases.

In this project, the FSM is designed around 3 primary states: The *Search* state, *approach* state, and *interact* state, as illustrated in Figure 5.2. The idea behind these states is as follows: When no object is detected, it operates in the *search* state. In this state, the robot actively scans the environment until it detects an object. Once an object is detected, the FSM transitions to the *approach* state, where motor control is used to move the robot towards the detected object based on an estimated distance. When the object reaches a predefined proximity threshold, it will change into the *interact* stage, and its focus will shift to controlling the arms. While this functionality is not explicitly implemented in the current system, the state is included in the FSM design to support such an extension. From the interact state, the system can transition back to the *search* state when a new object is to be interacted with, forming a closed control loop.

The modular design of the FSM provides flexibility and scalability. For instance, the interact state could easily be subdivided into additional substates, such as Arm Movement and Hand Movement, to support more complex manipulation behaviors in future extensions of the system.

5.2.2 Multithreaded Perception - Reflections

Multithreading in Python provides a way to structure a system into several concurrently running components within a single process. In principle, this can be advantageous for robotics applications, where different sub tasks such as image capture, object detection, depth estimation, data fusion, and movement control could be executed in parallel. Early in the development of this project, the intention was to follow this approach; separating the perception pipeline into multiple threads so that object detection and depth estimation could run simultaneously. Conceptually, this would reduce overall latency per frame and increase responsiveness, since both models are computationally demanding and operate on the same incoming image.

However, during implementation and testing, several practical issues arose when trying to use multithreading. While it improved structure and modularity, it also introduced unexpected performance penalties on the object detection model when combined with GPU-accelerated models. The suspected culprit was Python's threading model and the way CUDA contexts are handled: when multiple threads attempted to perform tasks involving the GPU, the YOLO model appeared to be scheduled or synchronized in a way that caused inference to slow down dramatically. An additional factor may have been related to the configuration of the Docker container, which could have influenced how GPU resources were shared or initialized between threads Ultimately, the supposed performance benefits of multithreading were outweighed by the resulting overhead, especially on the GPU.

This led to a shift in the system design. Instead of parallelizing the perception modules,

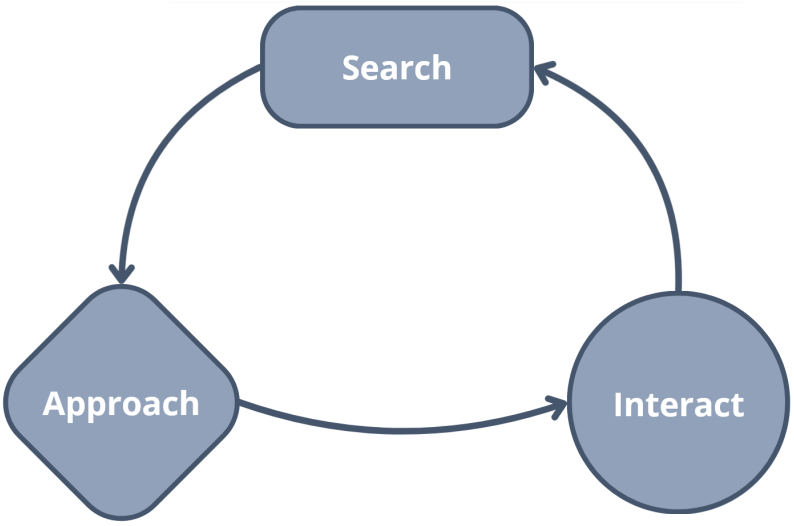


Figure 5.2: Simple figure illustrating how the different states in the system interacts in a loop.

the final implementation performs object detection and depth estimation sequentially in the main thread. Since they are not always running at the same time (e.g. depth estimation is not run in search state), the sequential pipeline remains fast and predictable. This also ensures that inference runs at full speed without interference from other threads.

Multithreading is still used in the system, but only where it is both safe and meaningful: for camera input. Running the frame capture in a separate thread ensures that new frames are continuously buffered, even while the main thread is busy performing model inference or updating the control state. This avoids blocking on Input/Output and prevents the robot's camera stream from becoming a bottleneck. Importantly, this thread performs no GPU operations and, therefore, should not trigger the performance issues encountered earlier.

Although the final system uses a largely single-threaded perception pipeline, this decision does not diminish the potential value of multithreading. If future work were to integrate additional models – such as tracking, pose estimation, or semantic segmentation – or if multiple GPU-accelerated modules needed to run concurrently, a more sophisticated parallel architecture might become necessary. In such cases, using multiprocessing (rather than threading) would likely be the preferred approach, as it avoids the CUDA-related limitations encountered in this project.

Overall, this experience highlights the difference between theoretical design advantages and practical performance on real hardware, and demonstrates the importance of validating architectural decisions experimentally.

5.3 Implementation Details

In this section, a more thorough description of each part of the system is provided. While the previous sections focused on the concepts and overall design, this section explains how these ideas are realized in practice through code. The complete implementation is publicly available in a GitHub repository[1], which also guides how to use the system locally.

[1] https://github.com/Feen-X/PepperMDE

5.3.1 Runtime Environment and Dependencies

To make the code less dependent on the individual hardware and software specifications, it was built to run inside a Docker environment that has all dependencies and necessary libraries bundled together into a single, reproducible runtime environment. In particular, Docker simplifies the deployment of the deep learning frameworks used, and the communication between them and the Pepper robot.

It is necessary to specify that while the YOLO-model is supposed to be downloaded beforehand and put into the correct folder, the Depth Anything V2 model will be automatically downloaded and cached locally when running the system for the first time. This initial download may take several minutes, but after both models are cached, subsequent executions of the system run without any additional overhead. Aside from GPU availability, which affects performance, the system should be able to be executed consistently across different machines.

5.3.2 System Organization and Modularity

The system is implemented primarily in Python and is organized around a single main script, with several other files providing helper functions and classes to it. The code is written with the purpose of being easy to modify, and simple to understand and debug. For this reason, the system can be executed in *two distinct modes*. The primary mode connects directly to a Pepper robot, streaming camera input from the robot to a local port and sending motion commands in real time. For debugging and experimentation without physical hardware, a *secondary image-based mode* is provided. In this mode, it runs the code on a variation of demo images that you can switch between, emulating the perception pipeline without issuing any robot commands.

The execution mode is controlled by a single boolean variable, `IMG_MODE`. When set to `True`, the system runs independently of the robot, being able to cycle through the demo images. When set to `False`, the system connects to the Pepper robot and operates in real time. This design allows rapid iteration during development while preserving identical control logic across both modes.

5.3.3 Main Loop Overview

The main loop acts as the central coordinator for the system and is responsible for switching between states during execution. Before entering the loop itself, the system performs all necessary initialization steps. This includes loading the object detection model (YOLO) and the monocular depth estimation model (Depth Anything V2). After this, a separate camera thread is started, which continuously captures frames from the Pepper robot and places the most recent ones into a queue, ensuring that the main loop always has access to up-to-date visual input.

Once the perception components are initialized, the finite-state machine is set up and the system enters its continuous execution loop. In the loop, three main things happen: First, the latest frame is retrieved by the queue (or, in image mode, from the set of static images). Second, the YOLO model is used to try to detect any objects. And finally, the data is then processed according to the current state of the system.

As mentioned, there are three distinct states: *Search, Approach* and *Interact*. Each state is implemented as a dedicated function, encapsulating the logic associated with that behavior and returns a string specifying which state the system should transition into. Depending on the situation, this may either trigger a state transition or keep the system in the same state. Currently, the system is implemented in a way so that states can only transition in the order shown by the arrows in Figure 5.2.

The main loop continues to run until it is explicitly terminated by a user command. Upon termination, the system performs a clean shutdown by stopping the camera stream, putting the Pepper robot to sleep, and safely disconnecting from it before exiting.

5.3.4 Search State Implementation

The behavior of the search state depends primarily on whether any objects are detected by the object detection model. Since the model may detect multiple of the same object class in a single frame, the system selects one object to focus on. This is done by identifying the most confident detection of the target object, which is then treated as the candidate for further action.

While no target object is detected, the robot actively scans its surroundings. This is achieved by sweeping the robot's head from left to right to cover a wider field of view. If the target object is not detected for an extended period of time, the robot additionally rotates its body by a fixed angle before continuing the head sweep. This combination of head movement and body rotation allows the robot to gradually explore the environment without remaining stationary.

Once the target object is detected, the search behavior changes. The robot stops the head sweep and turns its body such that the detected object is in the center of the camera image when the robot is facing forward. This alignment step simplifies the subsequent approach phase, as the robot can move toward the object without needing to correct its orientation continuously.

5.3.5 Approach State Implementation

In the approach state, it is assumed that the robot is already roughly facing the target object as a result of the search behavior. Therefore, it will start by applying the MDE model on the image to estimate the distance to the detected object. This is done by computing the median depth within the region of interest specified by the object's bounding box. This is the state that was illustrated earlier in Figure 5.1.

Based on this estimated distance, the robot then moves forward toward the target. To avoid collisions, a constant offset is subtracted from the estimated distance. While approaching, the system continuously checks whether the object is still visible in the frame, and keeps re-evaluating the depth to it, to make it more robust against noise or other errors.

The system remains in the approach state until one of two conditions is met. If the estimated distance is within a predefined threshold, the robot is considered close enough to the object, and the system transitions to the interact state. Alternatively, if the target object is temporarily lost during the approach, the system assumes that the object is located at the last detected position and continues moving forward until that distance is reached. Once this movement is completed, the system also transitions to the interact state.

This state was implemented with the primary goal of evaluating whether monocular depth estimation provides sufficiently reliable distance information for basic robot navigation. As a result, the control logic is intentionally simple and relies on direct threshold-based decisions rather than continuous control. While this is sufficient to demonstrate the feasibility of integrating MDE into the perception–action loop, it does not aim to achieve smooth or dynamically optimal motion.

For future work, the approach behavior could be refined by introducing more advanced control strategies based on control theory, enabling smoother and more adaptive motion toward the target.

5.3.6 Interact State and Extensibility

Currently, the interact state is implemented as a proof-of-concept. When this state is reached, the robot is programmed to acknowledge that it has arrived at the target object and then cycle the target to another object class before returning to the search state. This allows the full perception and navigation pipeline to be demonstrated without committing to a specific interaction behavior.

The intended purpose of the interact state is to handle direct interaction with the object that has been approached, such as grasping a drawer handle, pushing a chair, or performing other task-specific actions. Due to time constraints, implementing such behaviors was considered outside the scope of this project.

Importantly, the interact state is implemented as its own state, separating it from the other two. This separation makes the system easy to extend, as additional interaction logic can be added without modifying the rest of the pipeline. As a result, the current implementation serves as a flexible foundation for future work, where interaction behaviors can be integrated with minimal changes to the overall system.

5.4 Integration With Pepper

With the last Pepper robot being produced in late 2020 [45], the available resources and development support for the platform has have been sparse in the following years. Communication with the robot in this project is handled through the *qi* framework developed by Aldebaran Robotics, a company under SoftBank Robotics. This library was chosen as it is compatible with Python 3, allowing it to be used alongside the other libraries employed in the system.

The initial Pepper communication pipeline was developed with inspiration from the DTU course *02182 Symbolic Artificial Intelligence* (Spring 2025), which provided a general framework for connecting to Pepper and issuing high-level commands. However, the vision-related components of this framework proved to be unstable in practice, leading to frequent crashes and unreliable behavior. As a result, the vision part of the pipeline was substantially reworked. The revised implementation enables reliable image streaming and perception without disrupting the overall system execution with sudden crashes.

The use of the *qi* module as well as the pipeline from the DTU course was also part of the reason to develop the project inside a Docker container. The module relies on specific system dependencies and library versions, so by creating a container, it ensures that these dependencies are properly handled.

Through the *qi* API, the system uses Pepper's vision, motion, and speech modules. The vision module is used to retrieve the live video stream from the robot, while the motion module enables head movement, body rotation, and forward motion. The speech module is used primarily for debugging and status feedback during development. Pepper is equipped with two 2D cameras, of which only one is used in this project.

A significant bottleneck encountered during development was the performance of the onboard camera. Sometimes the frame-rate dropped to as low as 1-2 FPS, which noticeably slowed down the perception states and, consequently, the robot's responsiveness. Despite this limitation, the onboard camera was being used to maintain a fully self-contained system that relies only on the built-in hardware of the robot.

Thus, as a direction for future work, the system could be extended to use an external camera mounted on the robot, providing a higher and more stable frame-rate. Due to the

modular structure of the code, such a change would primarily only require replacing the video stream source.

5.5 Experimental Setup

The main components of the system and its integration with the Pepper robot have now been covered. Together, these enable Pepper to perceive its surroundings in real time and to act upon objects in the environment. This closes the perception–action loop and serves as a proof-of-concept for using monocular depth estimation as part of a robotic control pipeline.

To evaluate the robustness of monocular depth estimation within this perception–action framework, a simple experimental setup is used: Pepper is placed in an indoor environment that contains three different objects positioned at varying distances from the robot in a triangle. The task of the system is to identify a specified target object, approach it until a predefined distance is reached, and then switch its attention to the next object. Repeating this process causes the robot to move through the environment and sequentially approach each object.

The selected objects differ in size, shape, and visual complexity in order to challenge different aspects of the perception pipeline. The first object, a chair, represents a relatively simple and medium-sized target that is commonly present in indoor environments. The second object is a laptop (that is placed on a table), which is significantly smaller and contains finer visual details, which tests the depth estimation performance on smaller regions. The final object is another Pepper robot, which serves as a more challenging target. While the system searches for the class *person*, the detected object is not an actual human but a humanoid robot. This introduces ambiguity in object detection, as the target may be detected with varying confidence. Additionally, the complex shape of the robot provides a more difficult depth estimation compared to the other objects. This object therefore also tests the overall robustness of the perception–action pipeline.

The experiment does not aim to quantitatively evaluate the depth estimation model against ground truth measurements, as many evaluations already exist prior to this project. Instead, it evaluates whether the estimated depth is sufficiently accurate to enable meaningful robot behavior. Specifically, the experiment tests if Pepper is able to approach different objects reliably using only MDE for depth, as well as how robust the whole system is and handles uncertainty or temporary loss of detections during motion.

By observing the robot's ability to consistently reach an appropriate interaction distance for objects of varying size and visual complexity, the experiment highlights both the strengths and limitations of monocular depth estimation when used in a closed-loop perception-–action system.

6 Evaluation

In this chapter, the behavior of the proposed perception–action system on Pepper is evaluated. Here, the focus is on assessing whether monocular depth estimation enables reliable object localization and approaching in a real-world setting.

6.1 Experimental Procedure

The experiments were conducted in an indoor room, under normal lighting conditions, with an as low as possible amount of obstacles other than the target objects. Three objects were placed in the room at different positions, arranged approximately in a triangular layout with side lengths of around 4 m.

Each experimental trial is defined as one complete cycle through the search, approach, and interact states for all target objects. The sequence of target objects was fixed as *chair*, *laptop*, and *person*. After interacting with one object, the system automatically switched to the next target and repeated the process.

The experimental trials was repeated multiple times with varying starting positions and rotations. This was done to observe how the system behaves under different starting conditions and to assess the consistency of the perception–action pipeline. Stopping distance thresholds were also tested to find the proper distance to stop before the object.

Example trials are provided as supplementary material in the project's *Github repository*, and a snapshot from it is shown in Figure 6.1.

6.2 Results

Across multiple trials, the system was able to successfully detect and approach the correct target objects without major errors. In general, the robot consistently transitioned through

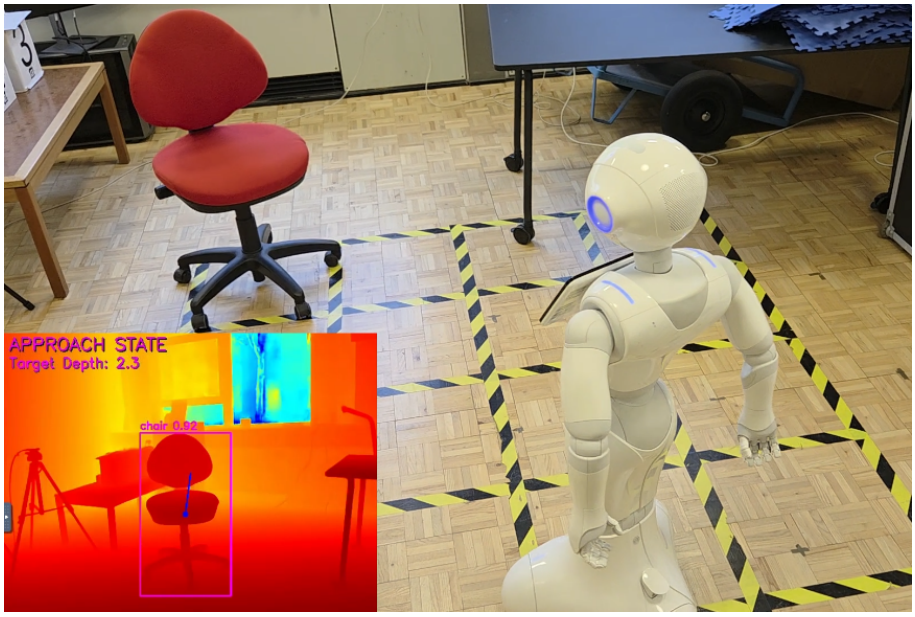


Figure 6.1: A frame from a trial example. Pepper is here in the *approach* state, targeting the chair in front of it. The lower left image shows what the system processes.

the search, approach, and interact states as intended, demonstrating that the perception–action pipeline functioned reliably in practice.

6.2.1 Behavior and Performance

In general, the system's estimated object distances were sufficiently accurate to enable a safe approach behavior. A stopping distance threshold of 1.25 m was chosen based on qualitative observation, as this value consistently resulted in the robot stopping within a reasonable interaction range. Lower threshold values, such as 0.75 m, occasionally caused the robot to move too close to the object, in some cases resulting in minor contact. This suggests that the effective error margin of the depth estimation is a little under 1 m.

The depth estimation proved robust across objects of varying size and shape. Large objects like the chair as well as the smaller laptop object were both approached successfully, indicating that the size of the region of interest did not significantly impact the quality of the depth estimates. The primary source of failure during the experiments was related to object detection rather than depth estimation. In particular, the detection of the *person* class was occasionally unstable, as the target object was another Pepper robot. In most cases, this issue was mitigated by the search behavior, which allowed the system to recover by continuing to scan the environment.

During the experiments, the FSM-based behavioral structure again proved beneficial, as the active state was always clearly identifiable, making the system easier to analyze and debug. Additionally, separating the logic into states ensured that only the relevant behavior was active at any given time, adding to a more clean performance.

6.2.2 Limitations

While the experiments demonstrated promising results, several limitations of the system became apparent. A major bottleneck was the low and inconsistent frame-rate of Pepper's own camera, as discussed in Section 5.4. In some cases, frames were delayed by multiple seconds, causing the system to react to outdated visual input, causing the reactions to be very slow. Testing across different Pepper robots showed variations in camera performance, but overall the limited frame-rate significantly reduced the responsiveness of the system.

Another relevant limitation to mention lies in the approach state. Although the robot continuously updates its depth estimates while moving toward the target, the current implementation does not account for obstacles between the robot and the object. Additionally, if the object moves horizontally during approach, the robot does not actively adjust its orientation to compensate, which may lead to suboptimal alignments.

Finally, the mentioned depth estimation error of approximately 1 m imposes constraints on interaction in close range. While sufficient for navigation and rough approach, this level of accuracy is not that adequate for fine manipulation that the interact state needs, where small distance differences become critical. Achieving more precise interaction would likely require specialized training or refinement of the depth estimation model for short-range distances.

6.3 Discussion

The experimental results effectively proved that it is feasible to construct a fully integrated perception–action pipeline for robotics using monocular vision alone, without relying on expensive depth sensors such as LiDAR or stereo cameras. Despite using a single 2D camera with limited resolution and low frame rate, the system was still able to estimate

object depth with sufficient accuracy to navigate a controlled environment and approach objects of varying size and shape.

These results support the findings presented in the survey in Chapter 4, which highlights how recent advances in monocular depth estimation have significantly improved robustness and generalization. The successful application of a modern monocular metric depth estimation model within a real robotic system illustrates how far the field has progressed to be used even on older robot platforms such as Pepper.

The primary challenge observed during the experiments was related to absolute depth estimation accuracy. Particularly at close range, the small errors in absolute distance were noticeable, even by using a state-of-the-art model such as Depth Anything V2. This behavior aligns with limitations also covered in the survey, where absolute depth from monocular input still remains an open challenge. Nevertheless, for tasks such as object localization, navigation, and coarse approach behavior, the achieved accuracy proved good enough.

Compared to other depth-sensing solutions, monocular depth estimation offers an accessible and cost-effective alternative, especially for simple platforms that do not need the integration of new hardware. While depth sensors provide higher accuracy at short distances, the results of this project suggest that monocular methods can enable meaningful perception–action capabilities when task complexity is moderate and the environment is controlled.

Overall, the project demonstrates that modern monocular metric depth estimation can serve as a viable foundation for vision based robotic interaction. Although limitations remain, the combination of object detection, depth estimation, and structured control logic successfully enabled the Pepper robot to perceive and act upon its environment, supporting the original problem formulation.

7 Conclusion and Future Work

In this project, monocular depth estimation was explored through two complementary approaches. First, an analytical survey examined the historical development of the field, outlining its evolution from traditional computer vision methods, through early deep learning approaches, to the current transformer-based era. This survey provided an overview of key paradigm shifts and highlighted recent advances that have significantly improved robustness and generalization.

Based on the insights gained from the survey, a state-of-the-art monocular depth estimation model, Depth Anything V2, was selected for practical evaluation in the second part of the project. This part focused on integrating monocular depth estimation into a perception–action system on the Pepper robot, with the aim of assessing whether such vision-based methods can support reliable object localization and approach behavior in a real-world robotic setting. The final, implemented system successfully demonstrated that a functional perception–action pipeline can be achieved using monocular vision alone, without relying on dedicated depth sensors.

While the system proved effective in controlled environments, the experiments also revealed limitations related to responsiveness and close-range depth accuracy. Despite these constraints, the results confirm that modern monocular depth estimation methods are capable of enabling meaningful robotic perception and navigation behaviors.

Overall, the survey and experiments together show that modern monocular depth estimation has matured to a level where it can support coarse navigation and object approach tasks, presenting it as an accessible and cost-effective alternative for robotic perception. Beyond the specific system implemented, the results indicate that monocular depth estimation can support a wide range of perception-driven robotic tasks, and is likely to play an increasingly important role as vision-based models continue to advance.

7.1 Future Work

Future work could extend the interaction state to enable finer-detail actions, further evaluating whether monocular depth estimation is sufficient for close-range interaction within a perception–action pipeline. Building on the existing finite-state machine structure, additional state logic could also be introduced to improve robustness, such as actively tracking objects that move horizontally away during the approach phase. Moreover, entirely new states could be implemented to enable, for example, obstacle navigation. Depending on the new states added, making the system parallel might be needed to keep it fast.

For tackling the perception challenges, the use of an external camera with a higher and more stable frame rate and upload stream could alleviate the limitations of the Pepper camera, enabling a more responsive system and allowing further investigation into the performance limits of monocular vision-based robotic control.

Finally, future work could explore alternative monocular depth estimation models. During the course of this project, the newer Depth Anything v3 model were published, and evaluating such models could provide insight into whether recent advances offer improved accuracy or robustness in a robotic control setting

Bibliography

[1] Jacqueline Du et al. *Humanoid Robot: The AI accelerant*. Tech. rep. Goldman Sachs, 2024. URL: https://www.goldmansachs.com/pdfs/insights/pages/gs-research/global-automation-humanoid-robot-the-ai-accelerant/report.pdf.

[2] Rasmus Ourø Lund. "Humanoide robotter på vej ind i industrien: »De kan agere i det rum, hvor hjul og automation kommer til kort«". Danish. In: *Ingeniøren* (2025). URL: https://ing.dk/artikel/humanoide-robotter-paa-vej-ind-i-industrien-de-kan-agere-i-det-rum-hvor-hjul-og-automation-kommer.

[3] Zhen Xu et al. *Towards Depth Foundation Model: Recent Trends in Vision-Based Depth Estimation*. 2025. arXiv: 2507.11540 [cs.CV]. URL: https://arxiv.org/abs/2507.11540.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. *Depth Map Prediction from a Single Image using a Multi-Scale Deep Network*. 2014. arXiv: 1406.2283 [cs.CV]. URL: https://arxiv.org/abs/1406.2283.

[5] SoftBank Robotics. *Pepper - Documentation*. 2024. URL: http://doc.aldebaran.com/2-5/home_pepper.html.

[6] Paul Viola and Michael Jones. "Rapid Object Detection using a Boosted Cascade of Simple Features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* (2001).

[7] Emily Philpot. *The Selfie Method: Exploring the mathematics behind your favourite Snapchat Lenses.* 2021. URL: https://emilyphilpot.medium.com/the-selfie-method-exploring-the-mathematics-behind-your-favourite-snapchat-lenses-aa7d3ca8d2f6 (visited on 11/28/2025).

[8] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640.

[9] Vrushali Pagire, Murthy Chavali, and Ashish Kale. "A comprehensive review of object detection with traditional and deep learning methods". In: *Signal Processing* 237 (2025), p. 110075. ISSN: 0165-1684. DOI: https://doi.org/10.1016/j.sigpro.2025.110075. URL: https://www.sciencedirect.com/science/article/pii/S0165168425001896.

[10] Shaobin Cai et al. "A human pose estimation network based on YOLOv8 framework with efficient multi-scale receptive field and expanded feature pyramid network". In: *Scientific Reports* (2025). ISSN: 2045-2322. DOI: https://doi.org/10.1038/s41598-025-00259-0. URL: https://www.nature.com/articles/s41598-025-00259-0.

[11] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics.

[12] Tianheng Cheng et al. "YOLO-World: Real-Time Open-Vocabulary Object Detection". In: *arXiv preprint arXiv:2401.17270* (2024).

[13] Glenn Jocher and Jing Qiu. *Ultralytics YOLO11*. Version 11.0.0. 2024. URL: https://github.com/ultralytics/ultralytics.

[14] Matthias Minderer et al. "Simple Open-Vocabulary Object Detection with Vision Transformers". In: *arXiv preprint arXiv:2205.06230* (2022).

[15] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[16] Xiaofeng Han et al. "Multimodal fusion and vision–language models: A survey for robot vision". In: *Information Fusion* 126 (2026), p. 103652. ISSN: 1566-2535. DOI:

https://doi.org/10.1016/j.inffus.2025.103652. URL: https://www.sciencedirect.com/science/article/pii/S1566253525007249.

[17] smbp. *RGB-D Images: A Comprehensive Overview*. 2024. URL: https://www.geeksforgeeks.org/computer-vision/rgb-d-images-a-comprehensive-overview/ (visited on 04/12/2025).

[18] Miles Hansard et al. *Time of Flight Cameras: Principles, Methods, and Applications*. SpringerBriefs in Computer Science. Springer, 2012, p. 95. ISBN: 978-1-4471-4658-2. DOI: 10.1007/978-1-4471-4658-2.

[19] Alberto Broggi. *A closer look at LiDAR and stereovision*. 2020. URL: https://www.ambarella.com.tw/blog/a-closer-look-at-lidar-and-stereovision/ (visited on 04/12/2025).

[20] A. Criminisi, I. Reid, and A. Zisserman. "Single View Metrology". In: *International Journal of Computer Vision* 40.2 (Nov. 2000), pp. 123–148.

[21] Ashutosh Saxena, Sung Chung, and Andrew Ng. "Learning Depth from Single Monocular Images". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper_files/paper/2005/file/17d8da815fa21c57af9829fb0a869602-Paper.pdf.

[22] Beyang Liu, Stephen Gould, and Daphne Koller. "Single image depth estimation from predicted semantic labels". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1253–1260. DOI: 10.1109/CVPR.2010.5539823.

[23] Ashutosh Saxena, Sung Chung, and Andrew Ng. "3-D Depth Reconstruction from a Single Still Image". In: *Int. Journal of Computer Vision*. 2008. DOI: https://doi.org/10.1007/s11263-007-0071-y.

[24] David Eigen and Rob Fergus. "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2650–2658. DOI: 10.1109/ICCV.2015.304.

[25] Iro Laina et al. *Deeper Depth Prediction with Fully Convolutional Residual Networks*. 2016. arXiv: 1606.00373 [cs.CV]. URL: https://arxiv.org/abs/1606.00373.

[26] Huan Fu et al. *Deep Ordinal Regression Network for Monocular Depth Estimation*. 2018. arXiv: 1806.02446 [cs.CV]. URL: https://arxiv.org/abs/1806.02446.

[27] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[28] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[29] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. *Vision Transformers for Dense Prediction*. 2021. arXiv: 2103.13413 [cs.CV]. URL: https://arxiv.org/abs/2103.13413.

[30] Ravi Garg et al. *Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue*. 2016. arXiv: 1603.04992 [cs.CV]. URL: https://arxiv.org/abs/1603.04992.

[31] Clément Godard et al. *Digging Into Self-Supervised Monocular Depth Estimation*. 2019. arXiv: 1806.01260 [cs.CV]. URL: https://arxiv.org/abs/1806.01260.

[32] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. *Semi-Supervised Deep Learning for Monocular Depth Map Prediction*. 2017. arXiv: 1702.02706 [cs.CV]. URL: https://arxiv.org/abs/1702.02706.

[33] Zhiwei Huang et al. “A systematic review of monocular depth estimation for autonomous driving: Methods and dataset benchmarking”. English. In: *Results in Engineering* 26 (June 2025). Publisher Copyright: © 2025 The Author(s). ISSN: 2590-1230. DOI: 10.1016/j.rineng.2025.105359.

[34] René Ranftl et al. *Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer*. 2020. arXiv: 1907.01341 [cs.CV]. URL: https://arxiv.org/abs/1907.01341.

[35] Reiner Birkl, Diana Wofk, and Matthias Müller. *MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation*. 2023. arXiv: 2307.14460 [cs.CV]. URL: https://arxiv.org/abs/2307.14460.

[36] Jiuling Zhang. *Survey on Monocular Metric Depth Estimation*. 2025. arXiv: 2501.11841 [cs.CV]. URL: https://arxiv.org/abs/2501.11841.

[37] Wei Yin et al. *Metric3D: Towards Zero-shot Metric 3D Prediction from A Single Image*. 2023. arXiv: 2307.10984 [cs.CV]. URL: https://arxiv.org/abs/2307.10984.

[38] Shariq Farooq Bhat et al. *ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth*. 2023. arXiv: 2302.12288 [cs.CV]. URL: https://arxiv.org/abs/2302.12288.

[39] Lihe Yang et al. “Depth Anything V2”. In: *arXiv:2406.09414* (2024).

[40] Maxime Oquab et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2024. arXiv: 2304.07193 [cs.CV]. URL: https://arxiv.org/abs/2304.07193.

[41] Bingxin Ke et al. “Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[42] Bingxin Ke et al. *Marigold: Affordable Adaptation of Diffusion-Based Image Generators for Image Analysis*. 2025. arXiv: 2505.09358 [cs.CV].

[43] Yiqun Duan, Xianda Guo, and Zheng Zhu. *DiffusionDepth: Diffusion Denoising Approach for Monocular Depth Estimation*. 2023. arXiv: 2303.05021 [cs.CV]. URL: https://arxiv.org/abs/2303.05021.

[44] Ming Gui et al. *DepthFM: Fast Monocular Depth Estimation with Flow Matching*. 2024. arXiv: 2403.13788 [cs.CV]. URL: https://arxiv.org/abs/2403.13788.

[45] Sam Nussey. *EXCLUSIVE SoftBank shrinks robotics business, stops Pepper production-sources*. 2021. URL: https://www.reuters.com/technology/exclusive-softbank-shrinks-robotics-business-stops-pepper-production-sources-2021-06-28/ (visited on 12/15/2025).

Technical
University of
Denmark

Richard Petersens Plads, Building 324
2800 Kgs. Lyngby

https://www.compute.dtu.dk/