DoggoBot
The Intelligent Dog Walking Robot
Project Report

MIE1075 Artificial Intelligence for Robotics I
Instructor: Prof. Andrew Goldenberg

Prepared by:

Chen, Ryan	Chen, Vicky	Wu, Yujie
1003912992	1004176795	1003968904

University of Toronto

January 9, 2023

Contents

1 Problem Statement

Dogs are the most loyal companions of human beings. They bring us happiness, entertainment, and a sense of safety. Raising dogs can effectively keep us away from loneliness and potential mental health issues, but it may also bring unexpected problems. Unlike other pets, dogs require their owners to take them for walks on a daily routine basis, not only to benefit their health and welfare, but most importantly, to allow them to excrete [1]. Smaller-sized dogs need at least one walk per day for at least 15 minutes, while larger dogs may require multiple [2]. However, not all pet owners are capable of regularly accomplishing this goal. Especially for students and busy workers, dog walking can be quite time-consuming yet mandatory.

1.1 Design Overview

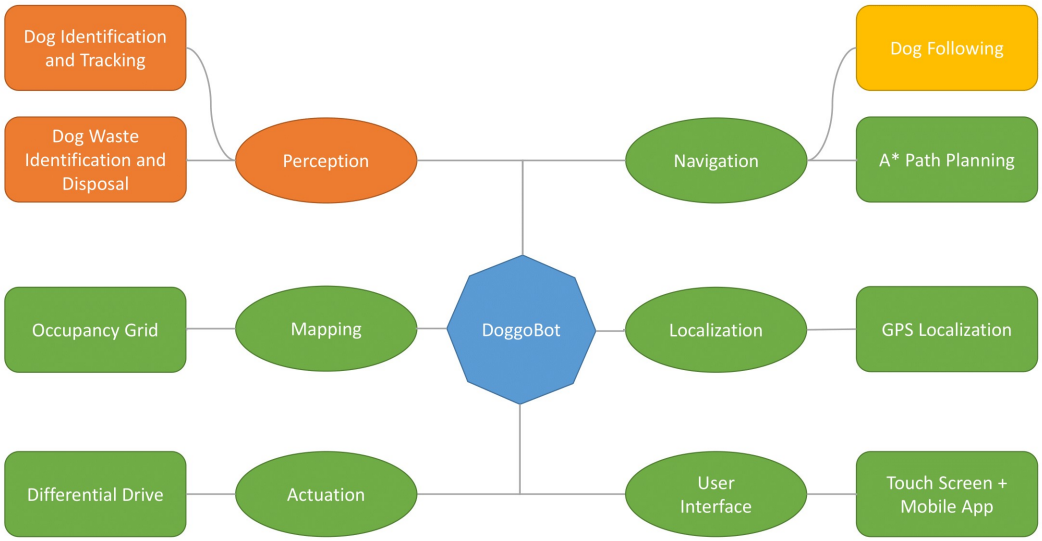

Figure 1: System architecture of DoggoBot. The oval shapes indicate the modules or subsystems that DoggoBot consists of. The rounded rectangle shapes highlight the specific methods/solutions utilized within each module. The orange bubbles specify the primary focus of the design project. The yellow bubble highlights elements of the design that are also investigated using AI-based solutions in addition to the focus of the project. The green bubbles illustrate the design aspects that can be completed using existing off-the-shelf solutions.

To address the above issue, we propose an intelligent dog walking robot, or DoggoBot, to bring an automated and robotized alternative to dog walking. DoggoBot is designed to be a differential drive mobile robot with several hardware features implemented. First, an extendable dog leash is attached to DoggoBot's frame to avoid the dog from running too far. Next, a camera is installed at the front of DoggoBot to enable a wide range of perception-based tasks. Lastly, a serial robotic arm with 3 degrees of freedom along with a container is installed to allow DoggoBot to perform customized operations, such as picking up dog waste.

To operate DoggoBot, we envision two operation modes, following mode and return mode, which can be selected and modified remotely through a mobile application interface. In following mode, DoggoBot will follow the dog and pick up any waste it may produce during the walk. In return mode, DoggoBot will plan the shortest path from its current location to the user's home. Then, it will take the dog back home by applying a force through the leash whenever the dog tries to venture off the planned path. In addition to these two modes, DoggoBot also has an emergency mode for suspicious and potentially dangerous activities. For example, if someone is trying to forcibly take the dog, the leash will be locked and an alert will be sent to the user immediately.

To enable these operation modes, DoggoBot also includes four key software modules that facilitate the fully autonomous dog walking experience, which are the perception, navigation, mapping, and localization modules. An overview of the system architecture of DoggoBot can be found in Figure 1. The perception module is responsible for tracking the dog and identifying dog waste in real-time using a learning-based approach. The navigation module is responsible for producing the dog following control outputs based on input from the perception module using a PID controller, as well as planning the shortest path from the current location to the desired destination through an A* path planning algorithm. The mapping module enables the local mapping of DoggoBot's surroundings to provide a detailed and up-to-date map for navigation using occupancy grids. The localization module utilizes GPS information to ensure a precise current position of DoggoBot on the map, which further facilitates an accurate navigation process. DoggoBot is able to report its real-time location obtained from the localization module to the user through an easy-to-use mobile application as well.

1.2 Design Focus

While DoggoBot consists of a variety of components as depicted in Figure 1, the primary focus of our design is the **perception module** of DoggoBot. Specifically, we aim to achieve learning-based real-time visual dog tracking and waste identification using convolutional neural networks (CNNs). In addition to these two tasks, we also explored the dog following aspect of the design as it is directly related to the perception module and requires the utilization of AI-based tools. Given the time constraints of the project, we recognize the remaining modules of DoggoBot as auxiliaries, where the components of these modules are non-AI-based and can be simply completed using existing off-the-shelf products and state-of-the-art solutions. In this report, we will only discuss the main focus of this project, which includes the tasks of **dog identification and tracking**, **dog following**, and **dog waste identification and disposal**.

1.2.1 Dog Identification and Tracking

As a dog walking robot, DoggoBot is designed to constantly monitor the dog's position using its front-facing camera. Such information is visually accessible through the user interface by observing bounding boxes around the dog. Moreover, the choice of tracking algorithm is robust and computationally inexpensive, as we anticipate the object to be tracked all the time and the tracking results should be consistent (i.e., no jumps between consecutive frames), while all computations are done in real-time. DoggoBot requires its user to identify a tracking target during initialization by selecting a region on the image, then it operates fully automatically.

1.2.2 Dog Following

The dog following task consists of two subtasks:

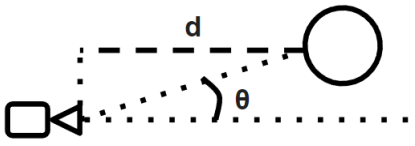

Figure 2: θ is the camera pose, representing the angle between the line of sight and the object. θ is used to determine the traverse direction of DoggoBot during the next control sequence. d is the distance between DoggoBot and the object, which is used as a reference for speed adjustment.

- **Steering Control by Tracking:** DoggoBot adjusts its driving direction based on the position of the object of interest. We designed a non-learning-based algorithm to perform camera pose estimation, as shown in Figure 2, which helps DoggoBot to aim at the object of interest.

- **Speed Control by Depth Estimation:** DoggoBot adjusts its driving speed based on its distance d from the object of interest, as shown in Figure 2. The depth map is estimated using input from DoggoBot's front camera at each frame, and the depth of the center pixel of the tracked object is taken as the estimated current distance from the object. The speed of DoggoBot is then controlled through a PID controller to maintain a constant distance from the object of interest. Given that the bottleneck of this process is the depth estimation component, extracting depth information from a monocular camera setup using a learning-based approach is the focus of this subtask.

1.2.3 Dog Waste Identification and Disposal

One important part of dog walking is allowing the dog to excrete waste. Dog waste is acidic and can be harmful to the soil and contaminate water if left unattended. Thus, DoggoBot should identify and pick up the waste produced by the dog promptly. The identification program will analyze real-time images taken by the camera and notify the robot as soon as it detects dog waste. The robot will then plan a path for the onboard manipulator to pick up the waste. During operation, the identification program should be efficient in the sense that it does not require a lot of computational resources. Additionally, the maneuver of the robotic arm should not cause any damage to surrounding animals, pedestrians, or plants when picking up and disposing the waste.

2 Background

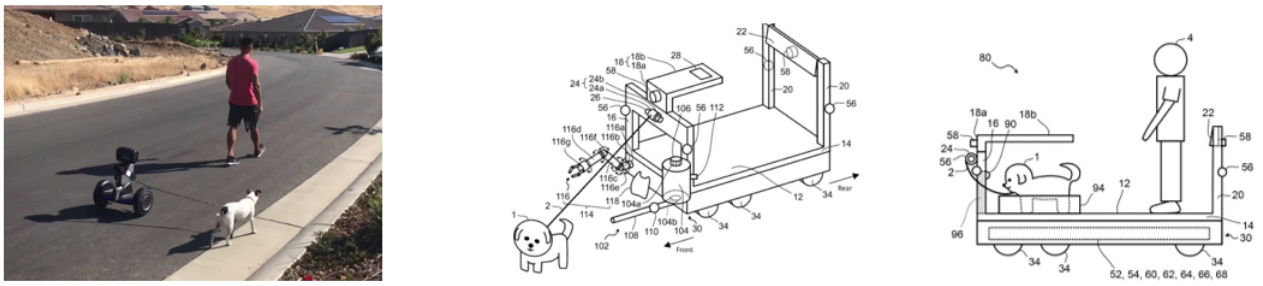

Figure 3: (a) A Loomo Puppy robot holding the dog leash while following the owner. (b) Schematic diagrams of the Toyota dog walking robot.

Our design draws inspiration from two sources. First is the Loomo Puppy robot proposed by Segway Robotics depicted in Figure 3(a). Loomo Puppy includes features such as dog face recognition, holding dog leash, dog voice control, and live GPS tracking [3]. However, it is not fully autonomous since it is unable to pick up dog waste nor can it handle dynamic situations such as dogs running out of control.

The second source of inspiration is a dog walking robot patented by Toyota shown in Figure 3(b). This autonomous robot is more sophisticated than Loomo Puppy, as it knows how to walk the dog along a pre-programmed route and pick up dog waste [4]. The robot also has a "Discipline Mode", which locks the leash if it detects the dog is trying to dash toward kids or squirrels. Additionally, it has a platform that allows the dog owner to ride along [4]. Nevertheless, the design of the robot is not agile and requires a significant amount of space.

Keeping these two designs in mind, a comprehensive literature review regarding the three main tasks previously presented in Section 1.2 is presented in the following sections.

2.1 Dog Identification and Tracking

The task of dog identification and tracking falls into the domain of arbitrary object tracking. Here we briefly go over some of the prior works in this field. Due to the absence of prior knowledge of the target's shape, the naive solution to arbitrary single object tracking is to develop task-specific models to learn the target's appearance using only data extracted from previous video frames, such as Kernelized Correlation Filter (KCF) [5] and P-N learning [6]. However, these approaches limit the complexity of the trained appearance models due to the lack of variety in demonstrations. Recent approaches to arbitrary single-object tracking aim to adapt deep convolutional architectures to this domain. With deep CNN, they no longer require individual models for each tracking task. For example, uniTrack [7] trains models through recent self-supervised learning techniques for a union of tasks, which allows the same appearance model to track different objects, and the models do not need to be trained in real-time (computationally inexpensive!).

2.2 Dog Following

2.2.1 Steering

Steering angle computation is essentially the problem of camera pose estimation. Conventional camera pose estimation tasks involve much more complicated components, such as detecting and matching image features. Prior works of camera pose estimation can be drawn from two categories: non-learning-based algorithms and learning-based architectures. Various non-learning techniques, such as SURF [8] and SIFT [9], are utilized to estimate camera poses, but these approaches lack the ability in handling complex background environments. CNNs are adopted to address such issues, but they require more complicated setups, such as stereo cameras for stereo video clips [10] and distance sensors to capture depth information [11]. Whereas, our problem is simpler. We only focus on horizontal pose estimation (as our robot can not move vertically) and we are aware of the position of the target. Thus, we chose to design a non-learning-based light pose estimation algorithm to save computation power for other tasks.

2.2.2 Depth Estimation

Depth estimation is the task of extracting depth information through perception to enhance the understanding of real three-dimensional scenes. Active depth estimation methods utilize lasers, light, and other types of reflections on object surfaces to establish depth points clouds and compute scene depth maps [12, 13]. However, obtaining such depth point clouds with high accuracy and density requires a significant amount of computation cost [14]. As a result, image-based methods have risen in popularity among the research community to tackle the depth estimation problem [15, 16]. Traditional image-based depth estimation obtains the depth map using a binocular camera setup, where stereo matching and triangulation are performed based on the disparity between the two 2D images [17, 18, 19]. Nevertheless, it can be difficult to capture a high number of distinct features to perform stereo matching when the scene has minimal textures. Furthermore, the requirement of two fixed cameras for such methods may not be feasible for all application scenarios [20]. Therefore, monocular depth estimation has become the focus of research in recent years [21, 22, 23, 24, 25]. One of the earliest investigations on learning-based monocular depth estimation was conducted by Eigen *et al.* [22], where a CNN was utilized to learn the desired depth map in a supervised learning manner. To eliminate the need for ground truth depth maps for training, Garg *et al.* [23] proposed an unsupervised learning approach, where the network learns depth information through an autoencoder network utilizing stereo image pairs. This work was further improved by Godard *et al.* through the introduction of a novel training loss that enforces depth consistency to produce the MonoDepth method [24]. For our problem, we chose to use MonoDepth2 [25], a new and improved version of the previously mentioned MonoDepth approach, to accurately extract depth information from DoggoBot's front camera.

2.3 Dog Waste Identification and Disposal

2.3.1 Object Detection

Object detection is a fundamental task for many computer vision systems and it involves detecting instances of objects in an image or video. The state-of-the-art approaches for object detection can be divided into two categories: one-stage methods and two-stage methods [26]. Some popular one-stage methods include YOLO, SSD, and RetinaNet. Models such as YOLO use CNN to produce predictions for regions across the entire image using anchor boxes [27]. While SSD focuses on applying small convolutional filters to feature maps to predict category scores and box offsets [28]. These one-stage models have fast processing speed with acceptable accuracy [26]. Unlike one-stage models, two-stage methods prioritize detection accuracy instead of processing speed. Models using the two-stage method include Faster R-CNN, Mask R-CNN, and Cascade R-CNN [26]. The first step of two-stage networks finds the region proposals or subsets of the image that might contain an object. Then the model classifies the objects within the region proposals during the second step [29].

2.3.2 Motion Planning for Robotic Arm

With the increased availability of robots, motion planning for robotic arms is becoming an important task. It is key to develop a method that can find an efficient and collision-free path for robotic manipulators, which allows the robot to safely interact with its surroundings while carrying out its tasks.

For robot motion planning, sampling-based methods such as Probabilistic Roadmaps (PRM) and Rapid Exploring Random Trees (RRT) find feasible paths for robot manipulators using graphs consisting of randomly sampled nodes and connected edges in the given configuration space [30]. More specifically, RRT builds a tree by sampling nodes from the neighbour of the starting node in the configuration space and grows the graph until the goal point is reached. Sampling-based methods are simple, fast, and easy to implement [31]. However, the paths they found are not guaranteed to be smooth. These methods also pose problems such as high memory usage and low search efficiency in a complex environment [32].

A traditional way of controlling the robotic arm is to use an artificial potential field. There are two kinds of artificial potential fields within the system: attractive fields and repulsive fields [33]. The attractive field encourages the end effector to move to the goal position, while the repulsive field helps the robotic arm move away from obstacles. In some cases, the artificial potential field approach is prone to local minima in the path planning algorithm, resulting in an inability to reach the target position [33].

Recently, deep learning-based control and operation of robotic manipulators have received a significant amount of attention. Reinforcement learning is a deep learning approach that finds an optimal policy [34]. The agent performs an action on the environment in accordance with the policy, and then the agent receives the next state and reward from the environment. The agent determines the best policy by maximizing the total rewards [34].

3 Methodology

In this section, the methods utilized to tackle the tasks outlined in the design focus (Section 1.2) are introduced. The integration procedure applied to produce a functional prototype of DoggoBot is provided as well.

3.1 Dog Identification and Tracking

To enable dog identification and tracking for DoggoBot, we follow the framework specified by Wang *et al.* [7]. This framework supports multiple tracking tasks, such as multiple object tracking, video object segmentation, and single object tracking. All of them share a similar workflow, we focus only on single object tracking in our work. The single object tracking branch involves two levels: a trainable base appearance model and a training-free propagation algorithm [7].

3.1.1 Appearance Model

Unlike prior arts, in which shallow models are trained for each specific object, Wang *et al.* [7] developed a procedure to train generalizable appearance models. First of all, the ultimate goal of this level is to extract latent features from the target object's shapes and appearances and encode them into feature maps $X = \phi(I) \in \mathbb{R}^{H*W*C}$. The choices of model architecture can vary from case to case. Theoretically, any deep CNN can be adopted, such as ResNets [35]. In practice, Wang *et al.* [7] selected the ones with small strides of $r = 8$ to accommodate the fact that the later object propagation task prefers high-resolution feature spaces to leverage fine-grained correspondences between frames. Self-supervision techniques are applied during the training phase to approximate pixel-level correspondences in non-synthetic datasets.

The network is optimized toward obtaining high similarity point vectors of $x_j^i \in \mathbb{R}^C$ across consecutive feature maps $x_k \in X, k \neq j$, while distinct from other non-related point vectors. Note the point vectors are extracted along the channel dimension C [7].

3.1.2 Mask Propagation

With the feature maps computed, a non-learning-based propagation algorithm is employed to estimate the target object's position (in the format of "masks") at each time step. Inspired by recent video self-supervised methods [36, 37, 38], Wang *et al.* [7] designed an affinity (propagation) matrix $K_{t-1}^t = [k_{i,j}]_{H*W*W*H}$ between consecutive frame X_{t-1} and X_t as following:

$$k_{i,j} = Softmax(X^{t-1}, X_t^T, \tau)_{i,j} = \frac{exp(sim(x_{t-1}^i, x_t^j)/\tau)}{\sum_k^{H*W} exp(sim(x_{t-1}^i, x_t^k)/tau)} \tag{1}$$

Where $sim()$ represents the inner product, τ is a hyperparameter to be tuned, and H and W represent the height and width of the frame respectively. The mask at frame t is predicted by propagating the mask at frame $t-1$ with the corresponding affinity matrix K_{t-1}^t. Note that for the initial frame, the mask is acquired from the ground truth.

With the above setups, the dog can be tracked by identifying target pixels on the masks at each frame.

3.1.3 Dataset and Experiments

Wang *et al.* [7] performed experiments with multiple self-supervised learning (SSL) techniques over the OTB dataset [39]. The OTB dataset contains 50 fully annotated sequences for various indoor and outdoor tasks with more than 600,000 frames. During all experiments, they deployed ResNet-50 [40] deep CNN as the architecture of the base appearance models. Area under curve (AUC) is selected as the evaluation metrics of the model performance, which measures the overlapping region between the tracked bounding box r_t and the ground truth bounding box r_{gnd}.

3.2 Dog Following

3.2.1 Steering

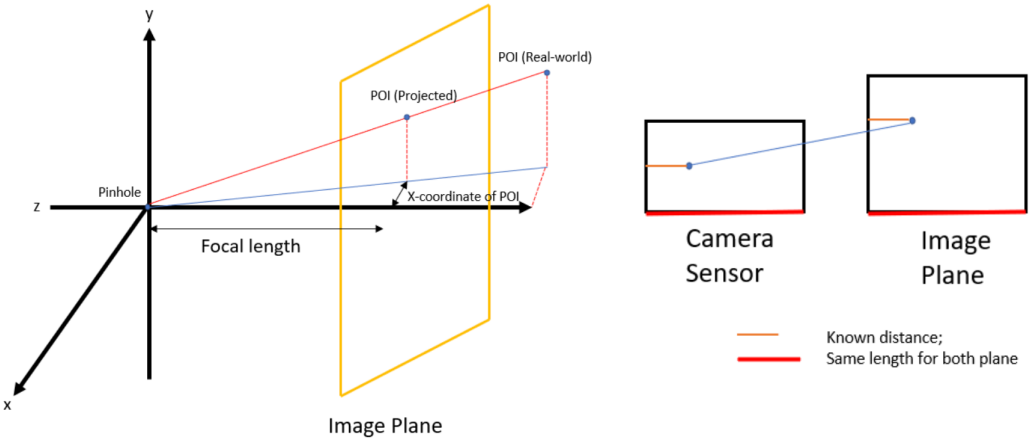

Figure 4: (a) A diagram of the basic pinhole camera geometry. Objects in the real world 3D space are projected onto a 2D image plane and captured by the camera sensor through a pinhole, which is a tiny opening that allows light to pass through. (b) An overview of the relationship between the image plane and the camera sensor. The image plane is formed by the union of coordinates in which lights emitted from these locations can pass the pinhole and fall on the camera sensor, thus their sizes are correlated.

Once the dog's position is monitored by DoggoBot, DoggoBot is able to adjust its steering angle to keep heading toward the dog. We designed a computational inexpensive steering angle adjustment algorithm based only on camera geometry. The computation of the steering angles takes a few steps:

- Extract the coordinates of the center of the object.
- Estimate the camera pose θ, as shown in Figure 2.

One approach to step 2 is to compute the tangent ratio between the focal length, which is available in the camera's intrinsic matrix, and the object's X coordinate on the image plane, as shown in Figure 4(a). However, the camera focal length is measured in millimetres, while the object's X coordinate is measured in pixel count. They can not be applied to tangent calculation directly. Given that the image plane is recorded by the camera sensor when shooting the photo, we can estimate the object's X coordinate in millimetres by some small tricks. According to similar triangles shown in Figure 4(b), the ratio between the camera sensor's length, which is also measured in millimetres, and the position of the object that appears on the camera sensor **equals to** the ratio between W and the object's X coordinate. Therefore, the position of the object that appears on the camera sensor can be easily computed, we estimate the object's X coordinate in millimetres by scaling up the figure by the focal length.

3.2.2 Depth Estimation

In order to obtain the necessary depth information to perform speed control, specifically the depth of the tracked dog, we chose to employ the MonoDepth2 method proposed by Godard *et al.* [25].

Based on the original MonoDepth method [24], MonoDepth2 uses an autoencoder network to perform unsupervised learning without the need for ground truth depth maps. This approach is achieved through the assistance of image reconstruction. Rather than giving the model a single image as input for training, a set of images are provided, where the model is trained to minimize the image reconstruction error based on depth extracted from a given image and projected onto the remaining views.

In the case of MonoDepth, the training input images are stereo image pairs, where the encoder produces a depth map prediction based on the left image. Using the obtained depth prediction and the right image, the decoder aims to reconstruct the left image with minimal errors [24]. However, while the trained encoder model can be utilized to produce depth maps from monocular images, the training process still requires stereo images, making the entire process quite counterintuitive. This is rectified through MonoDepth2, where the training input images are temporal frames from monocular videos instead of stereo image pairs [25]. This is accomplished through the introduction of a new pose estimation network, where the pose between the pair of frames is predicted. The additional pose information along with the predicted depth is utilized to perform image reconstruction.

In addition, MonoDepth2 introduces three architectural and loss innovations that significantly improve the accuracy of depth estimations [25], which include:

1. An appearance matching loss to address the problem of occluded pixels that occur when using monocular supervision.
2. An automasking approach to ignore pixels where no relative camera motion is observed.
3. A multi-scale appearance matching loss that performs all image sampling at the input resolution.

Dataset and Experiments: In the work by Godard *et al.* [25], this model was trained using the KITTI dataset [41] with the data split of Eigen *et al.* [42], which results in 39,810 sets of monocular frames for training and 4,424 for validation. Given that we do not possess an extensive outdoor vision dataset or any hardware with a significant amount of computing power, we chose to proceed with the pre-trained network by Godard *et al.* for our monocular depth estimation task.

3.3 Dog Waste Identification and Disposal

3.3.1 YOLOv5

YOLOv5 was selected to perform the dog waste identification task since it requires less computer memory, and is faster in processing time but remains similar accuracy as YOLOv4 on the same task. Additionally, YOLOv5 needs fewer training resources when compared to YOLOv7. Thus, YOLOv5 is a better choice for the object detection task.

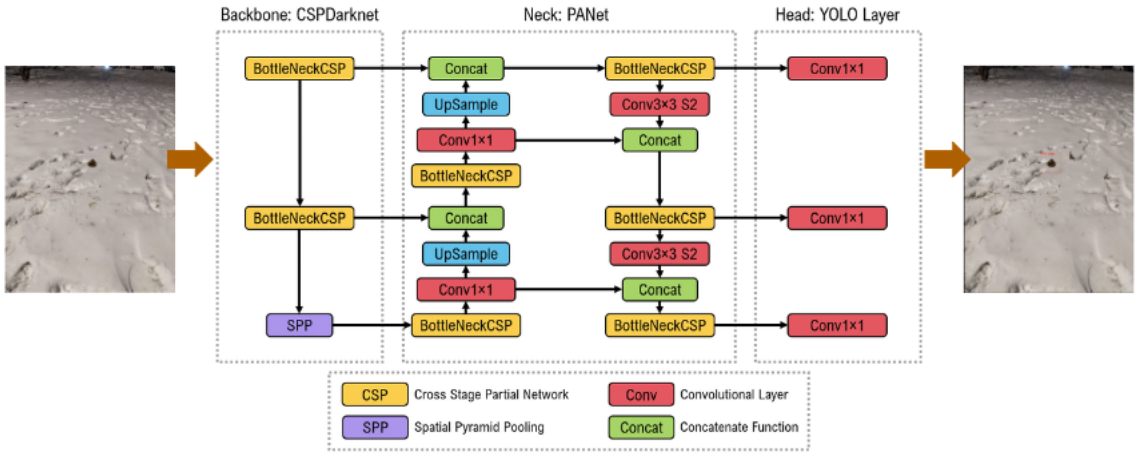

Figure 5: YOLOv5 Architectures. When an image gets pass into YOLOv5, the model backbone will first extract all important features from the image. Then, the model neck will generate feature pyramids to help with object identification. Lastly, the model head will produce predictions for bounding box coordinates, class probabilities, and objectness scores. If an object is identified in the image, a red bounding box will be drawn around the object.

YOLOv5 has 3 main parts: Backbone, Neck, and Head. The model backbone is mainly used to extract important features from the input image [43]. In YOLOv5, the Cross Stage Partial Network (CSPNet) is used as the backbone. CSPNet tackled the redundant gradient information problem which results in more efficient optimization and less costly inference computations. CSPNet has shown significant improvement in processing time with deeper networks [44].

Model Neck is mainly used to generate feature pyramids. Feature pyramids help to identify the same object with different sizes and scales which helps models to perform well on unseen data [43]. In YOLOv5 the Path Aggregation Network (PANet) is used. The PANet shortens the information path between layers which leads to better mask prediction [45].

The model Head is mainly used to perform the final detection. It applied anchor boxes on features and generates final output vectors. The final output vectors include predictions for bounding box coordinates, class probabilities, and objectness scores [46].

3.3.2 Dataset and Experiments

265 images were collected to train the dog waste identification model. 80% of the images were used for training (212 images) and 20% of the images were used for validation (53 images). We also set aside an additional 29 images for testing. The ground truth labels for all images are marked using the package Labelme. The photographs were taken by Jon Crall and the dataset contains a wide variety of background conditions that occur in upstate New York, including seasonal changes, snow, rain, daytime, nighttime, grass, and concrete [47]. Some examples of the images are shown in Figure 6.

Figure 6: Sample images from the dog waste identification dataset. The dataset contains images with various backgrounds. Here are four examples showing dog waste on snow, dog waste on grass, and dog waste surrounded by fallen leaves.

3.3.3 Dog Waste Disposal

A robotic arm is used to pick up and dispose the identified dog waste. We chose to demonstrate this operation using the KUKA robot in MATLAB simulation as shown in Figure 7. The KUKA robot has 6 revolute joints. It is an articulated manipulator with a three-degree-of-freedom wrist and a gripper.

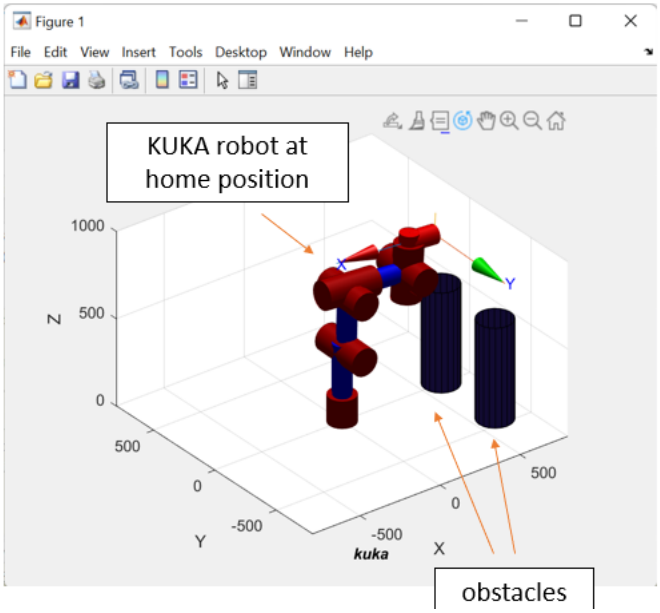

Figure 7: Dog waste disposal simulation setup in MATLAB. The KUKA robot has 6 joints and it is currently at its home position. Two cylinder obstacles are placed in front of the KUKA robot. The robot will need to move around while avoiding the two obstacles.

A motion algorithm involving artificial potential fields is used to control the robotic arm. The motion algorithm is based on iterations using gradient descent of a potential function that includes all attractive and repulsive potential functions as shown in Equation 2.

$$\boldsymbol{q}^{k+1} = \boldsymbol{q}^k + \alpha_k \sum_{i=1}^{n} \boldsymbol{J}_{\boldsymbol{v}_{o_i}}(\boldsymbol{q}^k)^T \left[\boldsymbol{F}_{\boldsymbol{i},\mathbf{att}}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}}(\boldsymbol{q}^k)) + \boldsymbol{F}_{\boldsymbol{i},\mathbf{rep}}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}}(\boldsymbol{q}^k)) \right] \tag{2}$$

where

$$\boldsymbol{F}_{\boldsymbol{i},\mathbf{att}}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}}) = \begin{cases} -\zeta_i(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \bar{\boldsymbol{o}}_{\boldsymbol{i}}^{\mathbf{0}}), & ||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \bar{\boldsymbol{o}}_{\boldsymbol{i}}^{\mathbf{0}}|| \leq d \\ -\zeta_i d \frac{(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \bar{\boldsymbol{o}}_{\boldsymbol{i}}^{\mathbf{0}})}{||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \bar{\boldsymbol{o}}_{\boldsymbol{i}}^{\mathbf{0}}||}, & ||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \bar{\boldsymbol{o}}_{\boldsymbol{i}}^{\mathbf{0}}|| > d \end{cases} \tag{3}$$

$$\boldsymbol{F}_{\boldsymbol{i},\mathbf{rep}}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}}) = \begin{cases} \eta_i \left(\frac{1}{||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \boldsymbol{\pi}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})||} - \frac{1}{\rho_0} \right) \frac{\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \boldsymbol{\pi}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})}{||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \boldsymbol{\pi}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})||^3}, & ||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \boldsymbol{\pi}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})|| \leq \rho_0 \\ 0, & ||\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \boldsymbol{\pi}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})|| > \rho_0 \end{cases} \tag{4}$$

$\boldsymbol{F}_{\boldsymbol{i},\mathbf{att}}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})$ and $\boldsymbol{F}_{\boldsymbol{i},\mathbf{rep}}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})$ are the attraction and repulsion forces. $\bar{\boldsymbol{o}}_{\boldsymbol{i}}^{\mathbf{0}}$ is the goal position of the end effector; $d > 0$ and $\rho_0 > 0$ are design parameters; The function $\boldsymbol{\pi}(\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}})$ is the orthogonal projection; ζ and η are parameters used to scale the effects of the attractive and repulsive potential.

Additionally,

$$\boldsymbol{J}_{\boldsymbol{v}_{o_i}} = \left[\boldsymbol{J}_{\boldsymbol{v}_1} \quad \cdots \quad \boldsymbol{J}_{\boldsymbol{v}_i} \quad | \quad \mathbf{0}_{\mathbf{3\times(6-i)}} \right] \tag{5}$$

and

$$\boldsymbol{J}_{\boldsymbol{v}_j} = \begin{cases} \boldsymbol{z}_{\boldsymbol{j-1}}^{\mathbf{0}} & \text{if joint } j \text{ is Prismatic} \\ \boldsymbol{z}_{\boldsymbol{j-1}}^{\mathbf{0}} \times (\boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}} - \boldsymbol{o}_{\boldsymbol{j-1}}^{\mathbf{0}}) & \text{if joint } j \text{ is Revolute} \end{cases} \tag{6}$$

The vectors $\boldsymbol{z}_{\boldsymbol{j-1}}^{\mathbf{0}}, \boldsymbol{o}_{\boldsymbol{i}}^{\mathbf{0}}$ and so on are computed using forward kinematics with joint vector $\boldsymbol{q}^k$. $\boldsymbol{q}^k$ is a column vector of current joint angles and $\boldsymbol{q}^{k+1}$ is the next angle position. $\boldsymbol{J}$ is the Jacobian matrix, and α is the step size. The gradient descent algorithm (Equation 2) generates a finite and discrete set of "waypoints" in the joint variable space. These finite sets of points help the robotic arm move from one position to the next position.

3.4 Integration

Given that all tasks related to the perception module of DoggoBot can be computed independently, the operation sequence of DoggoBot is as follows:

For each time step while the DoggoBot is in operation:

1. Track the dog by estimating its current position.
2. Adjust steering angle based on the dog's position.
3. Compute depth map of the visible scene.
4. Adjust speed based on the distance between the dog and DoggoBot.
5. Detect and pick up any waste produced by the dog.

To demonstrate the performance of the perception module, we implemented the previously described operation flow of DoggoBot on the DJI RoboMaster S1 robot to track and follow a person (as we do not have access to dogs to participate in our experiments). The DJI RoBoMaster S1 robot is equipped with 4 omni-directional wheels to allow more flexible steering controls. A 1080P camera is mounted at the top of the robot to capture the necessary video frames. Moreover, we handcrafted two pieces of "dog waste" with Play-Doh to demonstrate the ability of dog waste identification.

4 Results

In this section, the performance of each task outlined in the design focus (Section 1.2) is individually evaluated and discussed. Additionally, the performance of a functional prototype of DoggoBot produced through integration is reviewed as well.

4.1 Dog Identification and Tracking

4.1.1 Quantitative Results

Table 1: Tracking Performance (Partial) of Models Experimented by Wang *et al.* [7]

SSL Technique	Performance (AUC)
InsDis [48]	47.6%
MoCoV1 [49]	50.9%
MoCoV2 [50]	53.7%
PCLV1 [51]	**56.8%**
PCLV2 [51]	54.9%
ImageNet (Supervised)	**58.6%**

Table 1 illustrates the performance of models trained via various SSL techniques and a baseline (ImageNet) for comparison. Among all self-supervised techniques, the PCLV1 achieved the highest AUC score of 56.8% and was extremely close to the performance of ImageNet (58.6%), which was trained via a supervised fashion. Although non of the models trained via self-supervised techniques outperformed ImageNet, they were more generalizable in real-world scenarios since the constraint on synthetic datasets no longer exists.

4.1.2 Qualitative Results

Figure 8: Each row of the figure shows 5 consecutive frames from a video clip experimented by Wang *et al.* [7]. Both video clips contain completely different tasks. Specifically, the first video clip contains animals travelling in a unified direction on water. The second video clip depicts humans playing football on grass, where the movement directions of the players are varying. The tracking results for each frame are highlighted by green bounding boxes.

From the above video clips, we observe that the trackers precisely traced the motion of the targets. Moreover, they also demonstrated the ability of one unified appearance model for various kinds of object tracking. Moreover, the tracker also had satisfactory performance when handling complex background environments. In the second video clip, two players with nearly identical appearances were in close proximity to each other, whereas the tracker successfully distinguished them and did not lose its target.

4.2 Dog Following

4.2.1 Steering

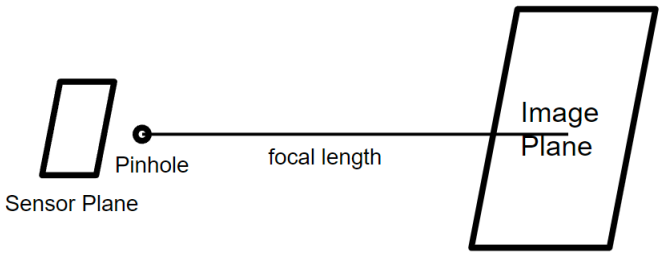

Figure 9: Simple demonstration of the spatial relationship between the sensor plane, pinhole, and the image plane. No direct contact between the sensor plane and the pinhole.

Since this algorithm is based entirely on camera geometry and no machine learning part is involved, we analyze potential sources of error within the algorithm in this section. First, we share some background information:

- Pinhole: a small hole that allows light (photons) to pass through.
- Focal length (f_x, f_y): the distance between the pinhole and the image plane. Normally, f_x and f_y are identical and can be found in the camera intrinsic matrix.
- Sensor plane: the plane where the camera sensor (device to capture the photons) is located.

The major source of error within this algorithm is the last step. Specifically, we scaled up the position of the object that appears on the camera sensor by the focal length when estimating θ. Indeed, the camera sensor is not directly attached to the pinhole, they are a few millimetres apart from each other, as shown in Figure 9. Therefore, when we scale up the figure by the focal length, which is measured by the distance between the pinhole and the image plane, we missed the distance between the camera sensor and the pinhole (which is not recorded by the camera intrinsic matrix and is extremely hard to find). Fortunately, they are close enough to each other, and omitting it will not result in a huge error. Moreover, as DoggoBot adjusts its steering angle at every single control step, DoggoBot is not likely to deviate too far from the target.

4.2.2 Depth Estimation

Table 2: Performance Comparison of MonoDepth2 to Existing Methods on the KITTI Benchmark [25]

Method	Train	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen [9]	D	0.203	1.548	6.307	0.282	0.702	0.890	0.890
Liu [36]	D	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Klodt [28]	D*M	0.166	1.490	5.998	-	0.778	0.919	0.966
AdaDepth [45]	D*	0.167	1.257	5.578	0.237	0.771	0.922	0.971
Kuznietsov [30]	DS	0.113	0.741	4.621	0.189	0.862	0.960	0.986
DVSO [68]	D*S	0.097	0.734	4.442	0.187	0.888	0.958	0.980
SVSM FT [39]	DS	0.094	0.626	4.252	0.177	0.891	0.965	0.984
Guo [16]	DS	0.096	0.641	4.095	0.168	0.892	0.967	0.986
DORN [10]	D	**0.072**	**0.307**	**2.727**	**0.120**	**0.932**	**0.984**	**0.994**
Zhou [76]†	M	0.183	1.595	6.709	0.270	0.734	0.902	0.959
Yang [70]	M	0.182	1.481	6.501	0.267	0.725	0.906	0.963
Mahjourian [40]	M	0.163	1.240	6.220	0.250	0.762	0.916	0.968
GeoNet [71]†	M	0.149	1.060	5.567	0.226	0.796	0.935	0.975
DDVO [62]	M	0.151	1.257	5.583	0.228	0.810	0.936	0.974
DF-Net [78]	M	0.150	1.124	5.507	0.223	0.806	0.933	0.973
LEGO [69]	M	0.162	1.352	6.276	0.252	-	-	-
Ranjan [51]	M	0.148	1.149	5.464	0.226	0.815	0.935	0.973
EPC++ [38]	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth '(M)' [5]	M	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Monodepth2 w/o pretraining	M	0.132	1.044	5.142	0.210	0.845	0.948	0.977
Monodepth2	M	**0.115**	**0.903**	**4.863**	**0.193**	**0.877**	**0.959**	**0.981**
Monodepth2 (1024 × 320)	M	**0.115**	**0.882**	**4.701**	**0.190**	**0.879**	**0.961**	**0.982**

In terms of quantitative results, the performance of MonoDepth2 is compared to existing methods on the KITTI benchmark shown in Table 2. Under the train column, D represents depth supervision, D* represents auxiliary depth supervision, and M represents unsupervised mono supervision. The best results in each category are in **bold**; the second best are underlined. It can be seen that MonoDepth2 outperforms all existing state-of-the-art unsupervised monocular depth estimation approaches across all metrics.

Figure 10: Depth map estimation of a sample outdoor test image with a dog in the frame.

As for qualitative results, the trained depth estimation network is used to predict the depth map of outdoor test images with dogs in the frames. An example image and its corresponding prediction output are shown in Figure 10. Given 50 similar test images, the depth of the dog can be clearly outlined in the output depth map for 90% of the images using the pre-trained MonoDepth2 network. Therefore, we believe this can be an accurate and reliable approach to obtain the current distance between DoggoBot and the dog to perform speed control.

4.3 Dog Waste Identification and Disposal

4.3.1 Quantitative Results

We trained the waste identification model for 280 steps. There was no significant improvement observed after step 280. Therefore, it is safe to stop training. We can see the training box loss is close to 0.02 and the validation box loss is close to 0.03. Both losses are quite low, which indicates the algorithm is able to identify the center of the object. The mean Average Precision (mAP) score is between 60% and 70% which shows the bounding boxes predicted by the model have acceptable performance.

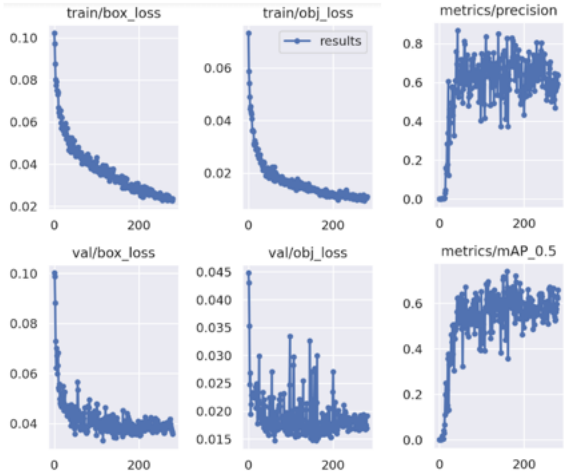

Figure 11: Training result for YOLOv5. The first column listed the training box loss and validation box loss. These two numbers reflect the accuracy of identifying the center of the object. The second column listed the training object loss and validation object loss. These two numbers reflect the accuracy of detecting objects within an image. The last column listed the precision and mAP. These two numbers reflect the quality of the predictions.

We also tested the model on 29 unseen images. We then divided the number of successful detections by the total number of dog wastes. The model successfully identified dog wastes in 27 images and achieved a ratio of 93.1%

$$\text{Ratio} = \frac{\text{Number of Successfully Identified Dog Waste}}{\text{Total Number of Dog Waste}} = \frac{27}{29} = 93.1\% \tag{7}$$

4.3.2 Qualitative Results

Figure 12: (a) Examples of successful dog waste detection. The images have various backgrounds. YOLOv5 successfully identifies dog wastes by drawing a red bounding box around each detection; (b) Examples of unsuccessful dog waste detection. No bounding box was drawn. YOLOv5 was not able to find dog waste within these two images.

Figure 12(a) depicts 3 examples of successful detections. The model works well with various background conditions. The first image was taken during winter and has snow on the ground, the next two images have grass. As we can see, the model has no problem identifying the dog waste in the images. Two unsuccessful detections are shown in Figure 12(b). In the first image, there is mud on the ground which makes the detection task difficult. The second image is also difficult because it has a relatively busy background with

trees, cars, and a garbage bin. Overall, YOLOv5 produces acceptable results and is suitable for the dog waste identification task.

4.3.3 MATLAB Simulation Results

To pick up the dog waste, we tested our idea using the KUKA robot in MATLAB simulation. In the simulation, we assume the position of the dog waste and the position of the obstacles are already known using the depth estimation model. We also abstract away the geometric detail of the obstacles, by assuming the shapes of the obstacles are convex sets. We wrote a motion algorithm to control the robotic arm to first pick up the dog waste and then move to the second location to dispose of it while avoiding two cylinder obstacles.

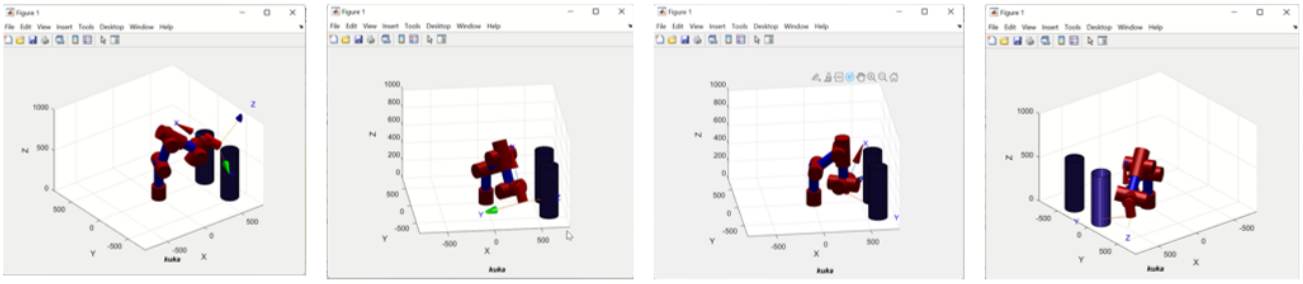

Figure 13: MATLAB video simulation of the dog waste disposal process. From left to right: 1) the KUKA robot moves from the home position to the first location to pick up dog waste; 2) The robot arrives at the first location; 3) Moves from the first location to the second location while avoiding two obstacles; 4) Arrives at the second location to dispose dog waste.

As shown in Figure 13, the robot is able to find a path from the home position to location 1 and then find a path from location 1 to location 2 without hitting obstacles or getting trapped in local minima.

4.4 Integration

Given this part contains the integration result combining all of the previously evaluated tasks, detailed experiments and quantitative performances for each task are available in their corresponding sections: 4.1, 4.2, 4.3. Additionally, we performed extra experiments to showcase some qualitative results of the integration process.

Figure 14: DoggoBot integration results. The first column displays the results from the tracking algorithm. Tracked objects are highlighted by white bounding boxes and tracking centers are annotated by blue dots. The middle column displays the depth map estimation, darker pixels correspond to closer objects. The last column displays the results from the waste identification algorithm. Detected wastes are labelled by red bounding boxes with confidence. No bounding box is present if there is no detection in the current frame.

Table 3: Steering Angle Estimation by Our Light Non-learning Based Algorithm

Row number	Steering angle
1	3.772°
2	-4.983°
3	-7.602°

We exported some processed frames during the operation of the robot, as shown in Figure 14. We also include the results from the steering angle computation in Table 3. In all plotted examples, the object was well tracked by the tracker and all wastes were detected by the robot. The steering angles were also within a reasonable range from visual inspection. However, we notice that the depth computation performed unexpectedly poorly. Due to the errors appearing on the depth maps, we decided to not control the robot's speed based on the depth map in the experiments. One explanation for the problem is that the dataset used to train the depth estimation model contains only outdoor images since the DoggoBot is designed for outdoor use only. Whereas the experiments we performed were purely indoors with highly identical walls and floors. The depth estimation model was not able to learn sufficient useful patterns and thus performed poorly.

5 Future Work

For future work, we can add a model that can understand dog postures. Currently, the algorithm performs waste detection in every frame. If there is a model that can understand dog postures, we only need to perform waste detection when the model detects the dog has just excreted waste. This way, our algorithm will become more efficient. Additionally, understanding dog postures will help the robot prepare for more dynamic situations. If the robot detects the dog is trying to dash toward kids or squirrels, it can respond by locking the leash in advance to avoid damage. One suitable model for tracking animal postures is

DeepLabCut. The model proposed by Mathis *et al* has been demonstrated to be versatile and has excellent animal tracking performance on test frames that is comparable to human accuracy [52].

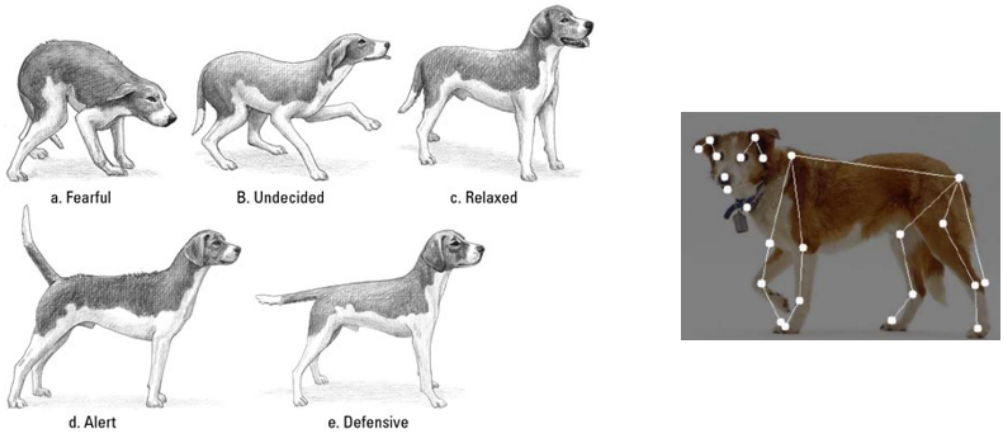

Figure 15: Understanding different dog postures by tracking key points on the dog.

Another area of improvement is motion planning for the robotic arm when picking up dog waste. Our current solution is a gradient descent algorithm, which is vulnerable to global minima. To improve, we can utilize more advanced algorithms and perform path planning using models such as RRT*.

Lastly, the integration experiments could be conducted under better testing conditions. The performance of the dog tracking, dog following, and dog waste disposal tasks could be more thoroughly evaluated with a real dog subject and outdoor scenes.

6 Conclusion

In conclusion, we proposed an intelligent dog walking robot, DoggoBot, to efficiently save dog owners' time by providing a fully autonomous dog walking experience. We have defined all necessary modules to form DoggoBot, with the main focus of this project being the perception module specifically. We managed to accomplish three major tasks related to the perception module, which include dog tracking, dog following, and dog waste disposal using state-of-the-art learning-based approaches with considerable performance. We believe that DoggoBot truly can be an automated and robotized alternative to dog walking in the near future.

References

[1] R. H. Kinsman, K. E. Main, R. A. Casey, R. E. Da Costa, S. C. Owczarczak-Garstecka, T. G. Knowles, S. Tasker, and J. K. Murray, "Dog walk frequency and duration: Analysis of a cohort of dogs up to 15 months of age," *Applied Animal Behaviour Science*, vol. 250, p. 105609, 2022.

[2] "How often should you walk your dog? here's what to consider," 2022. [Online]. Available: https://tractive.com/blog/en/good-to-know/how-often-should-you-walk-your-dog

[3] "Loomo personal robot," 2022. [Online]. Available: https://www.segway.com/loomo/

[4] "Toyota patents dog-walking robot and can pick up a pet's poop," 2022. [Online]. Available: https://www.motor1.com/news/579174/toyota-patents-dog-walking-vehicle/

[5] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 3, pp. 583–596, 2014.

[6] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2011.

[7] Z. Wang, H. Zhao, Y.-L. Li, S. Wang, P. Torr, and L. Bertinetto, "Do different tracking tasks require different appearance models?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 726–738, 2021.

[8] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[10] K. R. Konda and R. Memisevic, "Learning visual odometry with a convolutional network." in *VISAPP (1)*, 2015, pp. 486–490.

[11] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox, "Demon: Depth and motion network for learning monocular stereo," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5038–5047.

[12] G. Kim, B. Park, and A. Kim, "1-day learning, 1-year localization: Long-term lidar localization using scan context image," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1948–1955, 2019.

[13] T. Zhang, Y. Yang, Y. Zeng, and Y. Zhao, "Cognitive template-clustering improved linemod for efficient multi-object pose estimation," *Cognitive Computation*, vol. 12, no. 4, pp. 834–843, Jul 2020. [Online]. Available: https://doi.org/10.1007/s12559-020-09717-5

[14] J. Zhang, Q. Su, C. Wang, and H. Gu, "Monocular 3d vehicle detection with multi-instance depth and geometry reasoning for autonomous driving," *Neurocomputing*, vol. 403, pp. 182–192, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122030463X

[15] H. Luo, Y. Gao, Y. Wu, C. Liao, X. Yang, and K.-T. Cheng, "Real-time dense monocular slam with online adapted depth prediction network," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 470–483, 2019.

[16] J. Sun, Z. Wang, H. Yu, S. Zhang, J. Dong, and P. Gao, "Two-stage deep regression enhanced depth estimation from a single rgb image," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 719–727, 2022.

[17] J. Ren, A. Hussain, J. Han, and X. Jia, "Cognitive modelling and learning for multimedia mining and understanding," *Cognitive Computation*, vol. 11, no. 6, pp. 761–762, Dec 2019. [Online]. Available: https://doi.org/10.1007/s12559-019-09684-6

[18] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *CoRR*, vol. abs/1510.05970, 2015. [Online]. Available: http://arxiv.org/abs/1510.05970

[19] "Stereoscopic video saliency detection based on spatiotemporal correlation and depth confidence optimization," *Neurocomputing*, vol. 377, pp. 256–268, 2020.

[20] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian, "Monocular depth estimation based on deep learning: An overview," *Science China Technological Sciences*, vol. 63, no. 9, pp. 1612–1627, Sep 2020. [Online]. Available: https://doi.org/10.1007/s11431-020-1582-8

[21] T. Van Dijk and G. De Croon, "How do neural networks see depth in single images?" in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2183–2191.

[22] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2366–2374.

[23] R. Garg, B. V. Kumar, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*. Springer, 2016, pp. 740–756.

[24] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[25] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth prediction," October 2019.

[26] "What is object detection?" 2022. [Online]. Available: https://www.mathworks.com/discovery/object-detection.html#:~:text=Object%20detection%20is%20a%20computer,learning%20to%20produce%20meaningful%20results.

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

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Computer Vision – ECCV 2016*, p. 21–37, 2016.

[29] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[30] K. Gochev, V. Narayanan, B. Cohen, A. Safonova, and M. Likhachev, "Motion planning for robotic manipulators with independent wrist joints," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[31] D.-H. Kim, S.-J. Lim, D.-H. Lee, J. Y. Lee, and C.-S. Han, "A rrt-based motion planning of dual-arm robot for (dis)assembly tasks," *IEEE ISR 2013*, 2013.

[32] P. Liu, "Path planning of redundant manipulator based on improved rrt algorithm," *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, 2021.

[33] R. Siddiqui, "Path planning using potential field algorithm," Jul 2018. [Online]. Available: https://medium.com/@rymshasiddiqui/path-planning-using-potential-field-algorithm-a30ad12bdb08

[34] M. Kim, D.-K. Han, J.-H. Park, and J.-S. Kim, "Motion planning of robot manipulators for a smoother path using a twin delayed deep deterministic policy gradient with hindsight experience replay," *Applied Sciences*, vol. 10, no. 2, p. 575, 2020.

[35] B. Li, W. Wei, A. Ferreira, and S. Tan, "Rest-net: Diverse activation modules and parallel subnets-based cnn for spatial image steganalysis," *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 650–654, 2018.

[36] A. Jabri, A. Owens, and A. Efros, "Space-time correspondence as a contrastive random walk," *Advances in neural information processing systems*, vol. 33, pp. 19 545–19 560, 2020.

[37] Z. Lai, E. Lu, and W. Xie, "Mast: A memory-augmented self-supervised tracker," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6479–6488.

[38] X. Li, S. Liu, S. De Mello, X. Wang, J. Kautz, and M.-H. Yang, "Joint-task self-supervised learning for temporal correspondence," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[39] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[40] B. Koonce, "Resnet 50," in *Convolutional neural networks with swift for tensorflow*. Springer, 2021, pp. 63–72.

[41] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[42] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2650–2658, 2014.

[43] T. A. Team, "Yolo v5[U+200A]-[U+200A]explained and demystified," *Towards AI*, Jul 2020. [Online]. Available: https://towardsai.net/p/computer-vision/yolo-v5

[44] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[45] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[46] S. Vignesh, "Yolo: You only look once." Jun 2020. [Online]. Available: https://medium.com/analytics-vidhya/yolo-you-look-only-once-9af63cb143b7

[47] J. Crall, "Erotemic/shitspotter: An open source algorithm and dataset for finding poop in pictures." *GitHub*. [Online]. Available: https://github.com/Erotemic/shitspotter

[48] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[49] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[50] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[51] J. Li, P. Zhou, C. Xiong, and S. C. Hoi, "Prototypical contrastive learning of unsupervised representations," *arXiv preprint arXiv:2005.04966*, 2020.

[52] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, vol. 21, no. 9, p. 1281–1289, 2018.