Autonomous dozer sand grading under localization uncertainties

Yakov Miron[1,2], Yuval Goldfracht[1], Chana Ross[1], Dotan Di Castro[1] and Itzik Klein[2]
{Yakov.Miron, Yuval.Goldfracht, Chana.Ross, Dotan.DiCastro}@bosch.com, kitzik@univ.haifa.ac.il

***Abstract*— Surface grading, the process of leveling an uneven area containing pre-dumped sand piles, is an important task in the construction site pipeline. This labour-intensive process is often carried out by a bulldozer, a key machinery tool at any construction site. Current attempts to automate surface grading assume perfect localization. However, in real-world scenarios, this assumption fails, as agents are presented with imperfect perception, which leads to degraded performance. In this work, we address the problem of autonomous grading under uncertainties. First, we implement a simulation and a scaled real-world prototype environment to enable rapid policy exploration and evaluation in this setting. Second, we formalize the problem as a partially observable Markov decision process and train an agent capable of handling such uncertainties. We show, through rigorous experiments, that an agent trained under perfect localization will suffer degraded performance when presented with localization uncertainties. However, an agent trained using our method will develop a more robust policy for addressing such errors and, consequently, exhibit a better grading performance.**

I. Introduction

Recent years have seen a growing demand for automation at construction sites. First, automation can reduce the amount of manual labor required from construction workers, thus helping resolve the industry's labor shortage. Second, it can increase productivity, which has been stagnant in the last few decades, and cut down the inflating costs. Lastly, it can improve the workers safety by using machines on risky tasks.

Yet, incorporating automation into construction sites is a complicated endeavor due to the unpredictable, unstructured nature of these environments, where multiple machines and workers work simultaneously on a variety of challenging tasks. Construction projects are extremely varied, with each one tailored to a specific architectural design, specifications etc. For these reasons, automation in construction sites is an extremely difficult task, which has not yet been solved [1].

Recent advancements in artificial intelligence (AI), in the context of autonomous vehicles (AV), hold promise for automation in the construction industry. While indeed related, the automotive and construction industries nevertheless pose different challenges for automation. For example, data collection, which is the backbone of current methods for autonomous driving, is extremely challenging in the unstructured environment of construction sites, where safety, time, and cost are major practical considerations. This problem can be partially solved using simulators, but these, too, have their own set of drawbacks. In addition, the unpredictability of the construction environment, where extreme and dangerous scenarios happen frequently, has been found to be difficult to model and learn using standard methods for AV.

In this work, we address the problem of autonomous path planning for construction site vehicles. Specifically, our focus is on the autonomous grading task done by the bulldozer under localization uncertainty, where the estimated pose of the vehicle is erroneous. This task poses several challenges common to all machinery tools in any construction site. Therefore, the addressed problem can be considered a representative example in the field. The main challenges are data collection, which is a key difficulty for all machinery tools, partial observability of the environment due to sensor positioning, and sensory noise that causes localization uncertainty. The latter degrades agents perception and, thus, significantly impedes the decision-making process.

In order to overcome the difficulty of data collection, we use domain adaptation techniques [2] that can bridge the sim-to-real gap. In our approach, we augment the simulation so as to resemble as much as possible real-world data. We then train (and evaluate) a learned policy purely in simulation and test it in both simulation and in a scaled prototype environment. To overcome the localization uncertainty, we devise a novel training regime where the uncertainty is taken into account during the agent's policy training. This allows the agent to learn a robust policy, which improves its performance under uncertainty during inference compared to an agent trained in a clean, noise-less environment. Specifically, we augment the training dataset with many variations, including scale, rotation, and translation versions of the observation, thus improving the agent's ability to cope with more realistic scenarios where the observation is uncertain due to localization errors.

Our main contributions are as follows:

1) We show that an agent trained to perform a grading task in a perfect localization setting will under-perform when presented with real-world uncertainties.
2) We propose a training regime that takes the sensory noise into account to produce a robust policy in the presence of uncertainties.

We prove our hypotheses on both a simulation setup and a unique scaled real-world environment setup that includes a bulldozer with relevant sensors. The paper is organized as follows: Section II describes the related work, Section III depicts our hypotheses, formulates the problem and describes our training method. Section IV depicts the simulation pipeline, and real prototype environment results. Section V provides a conclusion and future work.

II. Related Work

A. Autonomous Vehicle Localization

Localization, i.e., estimating the current position and attitude of the vehicle with respect to its surroundings, is key to achieving safe, reliable and robust decision-making in autonomous driving (AD). While this subject has been extensively studied in the broad field of AD, it has received

far less attention in the context of autonomous construction vehicles. AD can be broadly split into two types of scenarios: driving in urban areas and on highways. Common approaches include *(a)* 3D registration-based methods, which fuse offline 3D maps to current Lidar scans [3]. *(b)* image-based methods, such as [4], which rely on image features to calculate the displacements between successive images. *(c)* deep learning-based methods, where images are used to predict the odometry using neural networks [5], [6]. In highway settings, the high velocity of the vehicle impinges the performance of the Lidar, [7].

Localization methods often rely on fusion of multiple noisy sensor measurements in order to produce a more accurate estimation of the vehicle's state. A common approach is the extended Kalman filter (EKF), a nonlinear version of the well-known linear Kalman filter, which linearizes about an estimate of the current mean and covariance. The EKF is considered the standard in the theory of nonlinear state estimation, navigation systems and GPS [8]. An extension of the EKF is error-state EKF (ES-EKF) [9], where the estimated state is the error between the current prediction and the measurement. Here, the errors have a less complex behaviour than the state itself and, therefore, the linearization is more accurate. In this setting, an aiding (primary) sensor produces fairly accurate but low-frequency measurements, given in the *navigation* coordinate system, is used alongside a high-frequency but less accurate sensor in order to produce a high-frequency and accurate estimation of the state [8].

B. Autonomous Vehicles' Decision-Making

Research in the field of decision-making for autonomous construction vehicles is limited and mainly focused on analysis of vehicle dynamics. An extensive analysis of the forces and moments applied to vehicles in off-road settings is presented in [10]. In contrast, research on autonomous vehicles in urban surroundings is abundant. The authors of [11] use a graph neural network (GNN) that exploits the spatial locality of individual road components, thus offering better scene understanding for decision-making. Moreover, [12] optimize driving comfort and travel duration while staying within the safety limits. In [13] data augmentation, i.e., add perturbations to the labeled trajectory, is applied as part of the training process. This method allows one to augment interesting scenarios, such as collisions and/or lane divergence, that are missing from the original labeled data. This process leads to a more robust policy that can handle diverse scenarios and overcome the distribution shift between the simulation and the real world. This approach proved to be efficient for agents' robustness to perception noise and was adopted in this work.

C. Sand Simulation

In the field of autonomous vehicles, great effort is devoted to the correct modelling of the interaction between the vehicle and its environment. Understanding and modelling this interaction is crucial for learning a good policy that will succeed in real-world settings. In the context of construction sites, in general, and bulldozers, specifically, a key factor in the behavior of the vehicle is its interaction with the soil, i.e., the sand is moved by the vehicle as it travels. Precise particle simulation can be used to model a vehicle's interaction with the sand; this approach involves using solid mechanical equations [14], discrete element methods [15], or fluid mechanics [16]. Recently, [17] utilized deep neural networks to simulate the reaction of particles to forces. While such methods might provide accurate modelling and realistic visualization, they often entail high computational costs. In the context of learning a policy, rapid data collection and evaluation are essential even at the expanse of simulation accuracy. We, therefore, must be able to capture the essence of the interaction, as doing so will allow the agent to succeed in the real-world without compromising the rendering speed. [18] established a simulator for earth-moving vehicles, that enables policy evaluation for bulldozers. This simulation was fairly accurate, fast and able to capture the main aspects of the interaction between the sand and the vehicle.

D. Autonomous Construction Vehicles

The area of autonomous construction vehicles has enjoyed a growing interest over the past few years. In this field, the majority of work has focused on bulldozers, excavators and wheel-loaders in multiple aspects of the autonomous driving task. [19] implemented a heuristic approach to grade sand piles. They examined the trade-off between grading the pile when the blade is in full capacity and pushing less sand to reduce the elapsed time. In [20] an A*-based path planning method for autonomous grading while taking into consideration the bulldozer's maneuvering capabilities was suggested. A ML approach for autonomous grading that exhibited better generalization capabilities in previously unseen scenarios, both in simulation and in real-world settings was presented in [18]. Moreover, [21] trained a privileged agent to mitigate the sim-to-real perception gap using a simulator. [22] developed a pioneering software package on a construction vehicle for automation; their excavator was one of the first to operate autonomously. The authors of [23] trained an agent using a model-free Reinforcement Learning approach. They relied on the PPO-CMA algorithm and a semi-recursive multi-body method and showed impressive results in simulation. Moreover, [24] offer a simulator for improving the performance and energy flow of wheel loaders. However, none of the presented methods takes into account the sensory noise of the system. This fact hinders the efforts to deploy these methods in a real-world environment. In [25] a MPC-based method for path following that is tailored for bulldozers was suggested.

E. POMDP

A partially-observable markov decision process (POMDP; [26]) consists of the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. The state $s \in \mathcal{S}$ contains all the information required to learn an optimal policy. However, agents are often provided with partial or noisy information regarding the environment, which is termed observation $o \in \mathcal{O}$. As opposed to states, observations typically lack the sufficient statistics for optimality. At each state s, the agent takes an action $a \in \mathcal{A}$. Then, the system transitions to the next state s' based on the transition kernel $P(s'|s, a)$. Finally, the agent is provided with a reward $r(s, a)$. The agent's goal is to learn a policy π that maximizes the cumulative *reward-to-go*, where the policy maps from the observations (or estimated states) to the actions.

F. Dataset Augmentation

Dataset augmentation is a common method and best practice for training ML models [27], [28]. If the model takes images as input, common augmentations often include basic image manipulations such as scale, rotation, translation, random erasing, color space changes (brightness changes etc.). In the context of decision-making, augmentations can help reduce the distribution mismatch between simulation and real-world samples [13]. Domain adaptation [2], [29] aims to improve the simulation's photo-realism in order to minimize the sim-to-real perception gap.

III. Autonomous Grading Under Uncertainties

In this work, we examine the effect of uncertainties in pose estimation on the performance of agents with respect to the grading task. In our approach, we model this uncertainty as noisy observations of the true state, as described in Section IV-A. Inspired by [13], we hypothesize the following:

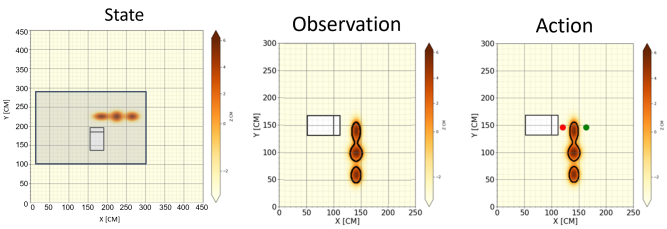
Hypothesis 1: *An agent trained under a perfect localization setting* $(agent_1)$ *will under-perform when presented with real-world uncertainties at inference.*

Hypothesis 2: *An agent trained with noisy observations* $(agent_2)$ *will develop a robust policy that can handle noisy observations at run-time.*

In this section, we discuss in detail the proposed method for training an agent under uncertainties for the purpose of learning a robust policy in real-world settings. Figure 1 provides an overview of our perception and decision-making pipeline, as described below.

A. Problem Formulation

In order to tackle the task of autonomous grading, we formalize it as a POMDP/R and define a 4-tuple consisting of states, observations, actions, and the transition kernel, as discussed in Section II-E. **States:** The state consists of all the information required to obtain the optimal policy and determine the outcome of each action. In our case, the state includes the accurate pose of the agent and the heightmap of the entire working area (Figure 2a). **Observations:** The observation is an image, as bounding-box representation of the full heightmap around the current location of the agent as shown in Figure 2a. As it aims to reflect the state's uncertainty caused by the sensor's noise, the inaccurate pose estimation translates into an erroneous bounding-box view around the bulldozer's location. In the simulation, we mimic this behavior by applying augmentation (rotation, translation) to the true, accurate state. Figure 2b is a case where an observation is derived from the true state without errors. **Actions:** An action is a pixel in the observation that the vehicle is requested to drive to. This pixel is connected to the physical world, assuming some pose information. We chose to focus on macro-actions [30], where the agent selects the destination coordinate and use classical methods for low-level control. We consider two aspects of the action in the context of errors: (i) *open-loop*, where the policy outputs a way-point for the agent to reach. Here, pose estimation errors are presented as a sub-optimal projection from state to observation. (ii) *closed-loop*, where errors in pose-estimation are fed back to the low-level controller for trajectory execution. Here, errors propagate through the system, leading to divergence from the desired path. Figure 2c includes examples of selected actions, in the form of way-points, shown as green and red dots. **Transitions:** Transitions are determined by a bulldozer's dynamics and the physical properties of the soil and the environment.



(a) (b) (c)

Fig. 2. Visualization of state, observation and actions. **(a)** The *full state*, where the agent has access to all the information, without errors. **(b)** *Observation*, where the agent has access to *part* of the information, which may include errors because the translation from state to observation (see section II-E) uses the estimated pose. The observation is derived from the grey rectangle of the state **(c)** *Actions*, i.e., the decisions made by the agent, given an observation, described as way-points, to reach and reverse (green and red dots respectively).

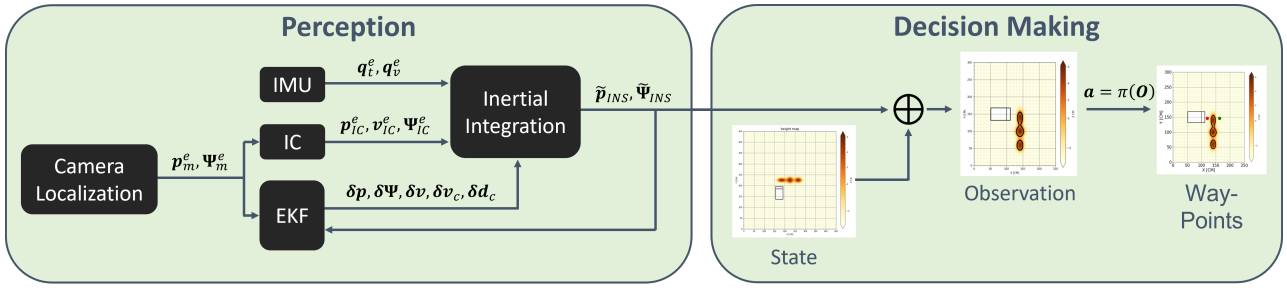


Fig. 1. Overview of the suggested method: **Perception**: An IMU provides velocity $\mathbf{q}_v^e$ and angular rate $\mathbf{q}_t^e$ increments at high frequency. The camera provides aiding position $\mathbf{p}_m^e$ and attitude $\mathbf{\Psi}_m^e$ measurements at low frequency. The inertial integration algorithm uses IC and IMU measurements to produce position $\tilde{\mathbf{p}}_{INS}$ and attitude $\tilde{\mathbf{\Psi}}_{INS}$ estimates at high frequency. The EKF uses the aiding measurements to correct the pose and bias estimates, which are are fed back to the inertial integration algorithm for increment compensation. The perception block outputs an estimate of the pose at high frequency. **Actuation**: The estimated pose from the perception block is fed to the simulator in order to render an observation from the true state. Once the observation is available, it is fed to the policy π, which provides way-point decisions. The simulator then performs these actions, and this process is repeated.

B. Training Under Uncertainties

In order to validate our hypothesises, we train *two* agents, each one under one of *two* noise settings: **Noise-free**: In this setting, we use the true trajectory, within the simulation, in order to extract noise-free observations of the current state. This scenario serves as a baseline for future comparisons. We consider the actions taken by the agent under this setting as the optimal policy. **Noise with Sensor Fusion Filtering**: In this setting, we render many noisy observations (augmentations) generated by the sensor fusion filtering algorithm. We do so by (i) adding synthetic noise to the inertial sensors, arranged in a typical inertial measurement unit (IMU), and to the aiding sensor measurements, (ii) applying our inertial navigation system (INS) and EKF equations as described in section IV-A and Figure 1, (iii) rendering the noisy observations from the distribution produced by the filter. These actions introduce uncertainties into our training pipeline. We denote the agents trained using the observations derived from the (i) noise-free and (ii) noise-with-sensor-fusion-filtering settings as $agent_1$, $agent_2$, respectively. In practice, inserting sensory noise into our measurements translates into small perturbations around the original observations. This process enhances our training dataset, which now includes a much wider distribution of potential states. This, in turn, allows the agent to learn a policy that is more robust to localization uncertainty. Figure 3 visualizes the augmented training dataset, where possible observations are rendered from the estimated pose $\tilde{\mathbf{x}}$, taking into account the pose uncertainty from the EKF covariance matrix estimate $\mathbf{\Sigma}$, i.e.:

$$\{\tilde{\mathbf{x}}_k\}_{k=0}^{K-1} \sim \mathcal{N}(\tilde{\mathbf{x}},\, \mathbf{\Sigma}) \tag{1}$$

Here, K is the number of observations that were rendered from this distribution about the estimated pose $\tilde{\mathbf{x}}$, and $\mathcal{N}(\cdot,\cdot)$ is the normal distribution,

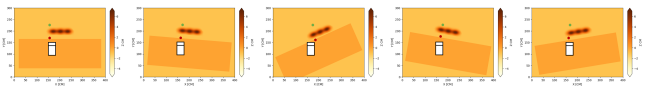


Fig. 3. Visualization of the augmented training dataset. The left image presents the true pose, i.e., the true observation. The green and red dots are the actions' training labels. The remainder images are rendered observations from the distribution of the EKF estimation, and their predicted training labels, in the form of way-points, shown as green and red dots.

C. Navigation Filter

An INS uses its inertial sensor readings and initial conditions (IC) to calculate the platform's position, velocity, and orientation [8]. In our implementation, we employ a sensor fusion approach, using EKF, between the IMU readings and an additional aiding sensor. When applying inertial integration to the IMU readings, we can neglect the effect of Earth's rotation, as the episode time is short [8]. Within the INS algorithm, we approximate the derivative as a temporal difference and, therefore, use Euler integration [31], rather than trapezoidal integration, which is commonly used in numerical integration [31] and inertial navigation [8]. In practice, we make use of the angular (2a) and velocity (2b) increments at time k from the inertial sensor.

$$\mathbf{q}_t[k] \in \mathbb{R}^{3\times 1} \tag{2a}$$
$$\mathbf{q}_v[k] \in \mathbb{R}^{3\times 1} \tag{2b}$$

In addition, we define the $E2D$ operator as follows:

$$\mathbf{D} := \mathbf{F}(\mathbf{q}_t[k]) = E2D(\mathbf{q}_t[k]) \in \mathbb{R}^{3\times 3} \tag{3}$$

This operator describes the deterministic transformation from Euler angels (ψ, θ, ϕ) to a transformation matrix $\mathbf{D}$, referred to hereunder as $\mathbf{F}(\cdot)$. The following equations describe our implementation of the inertial integration:

$$\mathbf{D}_n^b[k] = \mathbf{F}(\mathbf{q}_t[k]) \cdot \mathbf{D}_n^b[k-1] \tag{4a}$$
$$\tilde{\mathbf{v}}^n[k] = \tilde{\mathbf{v}}^n[k-1] + \mathbf{D}_b^n[k] \cdot \mathbf{q}_v[k] + (0,0,g\cdot dt)^T \tag{4b}$$
$$\tilde{\mathbf{p}}_{INS}^n[k] = \tilde{\mathbf{p}}_{INS}^n[k-1] + \tilde{\mathbf{v}}^n[k]\cdot dt \tag{4c}$$

In (4a), $\mathbf{D}_n^b[k]$ denotes the attitude transformation matrix between the navigation and body frames at time k, i.e., after integration and $\mathbf{D}_n^b[k-1]$ before integration. In (4b), $\tilde{\mathbf{v}}^n[k], \tilde{\mathbf{v}}^n[k-1]$ are the current and previous estimated velocities, respectively, $\mathbf{D}_b^n[k]$ is the transformation matrix from body to navigation frame, where $\mathbf{D}_b^n[k] = (\mathbf{D}_n^b[k])^T$, g is Earth's gravitational constant, and dt is the sampling frequency of the inertial sensor. In (4c), $\tilde{\mathbf{p}}_{INS}^n[k], \tilde{\mathbf{p}}_{INS}^n[k-1]$ are the current and previous estimated positions, respectively. Our sensor fusion algorithm relies on the error-state extended Kalman filter (ES-EKF) to fuse pose estimates from the INS integration algorithm with the low-frequency pose measurements from the aiding sensor (see section III-C). In simulation, our aiding sensor measurements are derived from the generated trajectory pose, which can be either noiseless or contain synthetically added noise. In our scaled real-world environment, the aiding sensor measurements are estimated using an ArUco marker [32] captured by our perception setup (see Figure 6). These measurements are inherently noisy and can, therefore, approximate a real-world experiment. In addition, we solely model the IMU's constant bias, neglecting other terms (e.g., in-run stability, as these terms are negligible for short time episodes). In our implementation, we define the following:

$$\delta\mathbf{x} = [\delta\mathbf{p}, \delta\mathbf{\Psi}, \delta\mathbf{v}, \delta\mathbf{b}_c, \delta\mathbf{d}_c]^T \in \mathbb{R}^{15\times 1} \tag{5a}$$
$$\delta\mathbf{z} = [\delta\mathbf{p}_m, \delta\mathbf{\Psi}_m]^T \in \mathbb{R}^{6\times 1} \tag{5b}$$

In (5a), $\delta\mathbf{x}$ is the error-state vector, $\delta\mathbf{p}$ is the position error, $\delta\mathbf{\Psi}$ is the attitude error, $\delta\mathbf{v}$ is the velocity error, and $\delta\mathbf{b}_c, \delta\mathbf{d}_c$ are the constant accelerometer and gyroscope bias, respectively. In (5b), $\delta\mathbf{z}$ is the difference between the measurement from the aiding sensor, provided by the perception setup, and the estimate from the INS algorithm, i.e., $\delta\mathbf{z}$ is the measurement residual. In our setup, the aiding measurement includes the position and attitude, where $\delta\mathbf{p}_m = \mathbf{p}_m^e - \tilde{\mathbf{p}}_{INS}^n$ is the positional measurement difference, $\mathbf{p}_m^e$ is the noisy position measurements from the aiding sensor, and $\tilde{\mathbf{p}}_{INS}^n$ is the noisy position estimate, according to (4c). $\delta\mathbf{\Psi}_m$ is the attitude measurement residual, according to [8]:

$$\delta\mathbf{\Psi_m} = \mathbf{h}(\mathbf{F}(\mathbf{\Psi}_m^e)^T \cdot \mathbf{F}(\tilde{\mathbf{\Psi}}_{INS})) \in \mathbb{R}^{3\times 1} \tag{6}$$

Here, $\mathbf{\Psi}_m^e$ and $\tilde{\mathbf{\Psi}}_{INS}$ are the noisy attitude measurements from the aiding sensor and INS algorithm, respectively, $\mathbf{F}(\cdot)$

is the $E2D$ operator defined in (3), and $\mathbf{h}(\cdot)$ is the $D2E$ operator, which represents a deterministic function from a transformation matrix $\mathbf{D}$ to Euler angels (ψ, θ, ϕ), defined as follows:

$$(\psi, \theta, \phi) := \mathbf{h}(\mathbf{D}) = D2E(\mathbf{D}) \in \mathbb{R}^{3\times 1} \tag{7}$$

$\tilde{\mathbf{\Psi}}_{INS}$ is derived when applying the $\mathbf{h}(\cdot)$ operator to $\mathbf{D}_n^b[k]$ in (4a). The intuition behind (6) is that the product of $\mathbf{F}^T \cdot \mathbf{F}$ matrices corresponds to a difference operator in linear (Euler angles) space. The system matrix of our ES-EKF implementation is as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{I}_3 \cdot dt_m & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & -\Sigma\mathbf{D} \\ \mathbf{I}_3 \cdot \frac{1}{dt_m} & \mathbf{A}_s & \mathbf{I}_3 & \Sigma\mathbf{D} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \in \mathbb{R}^{15\times 15} \tag{8}$$

Here, $\mathbf{I}_3, \mathbf{0}_3$ are a 3×3 identity and zeros matrix, respectively, $\Sigma\mathbf{D}$ is the sum of the rotation matrix between the measurement periods k and $k+1$, i.e., $\Sigma\mathbf{D} = \sum_{l=k}^{l=k+1} \mathbf{D}[l]$, $\mathbf{A}_s$ is a skew symmetric matrix representing the difference between two velocity measurements, and dt_m is the time interval between measurements k and $k+1$. In addition, as we update the position and attitude within the EKF, we define the following measurement matrix:

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \in \mathbb{R}^{15\times 6} \tag{9}$$

IV. Analysis and Results

A. Simulating Uncertainties

We used the simulation from [18] as a starting point, and made considerable modifications to it in order to support localization errors due to sensory noise. The simulation provides a full trajectory, i.e., positions and attitudes, at the beginning and end of each leg. In order to simulate the localization uncertainties in a realistic manner, we added the ability to generate a full trajectory at high frequency, e.g., 100 Hz. These high-frequency measurements allow us to simulate the sampling rate of the inertial sensor. From these high-rate measurements, we sample low rate, e.g., 1 to 10 Hz, aiding measurements required for the sensor fusion algorithm.

In practice, we generate a translation trajectory using a linear interpolation of the shortest straight path between the initial and terminal locations in a grading leg. Similarly, for a rotation trajectory, we interpolate between the initial and terminal attitudes of each leg (see Algorithm 1). We consider this trajectory a true trajectory executed by a bulldozer. We then apply a second-order derivative to the true positions and a first-order derivative to the attitude measurements in order to retrieve the velocity and rotation increments, respectively [8]. These measurements are considered the clean inertial readings (see Algorithm 2), and are given in body coordinates. Building off the clean inertial increments, we can then add sensor noise in order to simulate real measured inertial increments such as:

$$\mathbf{q}_v^e = \mathbf{q}_v^t + \mathbf{b}_c \Delta t + \mathbf{a}_{rw}\sqrt{\Delta t} \in \mathbb{R}^{3\times 1} \tag{10a}$$
$$\mathbf{q}_t^e = \mathbf{q}_t^t + \mathbf{d}_c \Delta t + \mathbf{g}_{rw}\sqrt{\Delta t} \in \mathbb{R}^{3\times 1} \tag{10b}$$

where $\mathbf{q}_v^e$ is the erroneous version of $\mathbf{q}_v^t$, $\mathbf{b}_c$ is the constant bias of the accelerometer, and $\mathbf{a}_{rw}$ is the accelerometer random walk. Similarly, $\mathbf{q}_t^e$ is the erroneous version of $\mathbf{q}_t^t$, $\mathbf{d}_c$ is the constant bias of the gyroscope, and $\mathbf{g}_{rw}$ is the gyroscope random walk. In addition, IC errors are applied to the true position and velocity vectors $\mathbf{p}_{IC}^t, \mathbf{v}_{IC}^t$, respectively:

$$\mathbf{p}_{IC}^e = \mathbf{p}_{IC}^t + \mathbf{\Delta p}_{IC} \in \mathbb{R}^{3\times 1} \tag{11a}$$
$$\mathbf{v}_{IC}^e = \mathbf{v}_{IC}^t + \mathbf{\Delta v}_{IC} \in \mathbb{R}^{3\times 1} \tag{11b}$$

where $\mathbf{\Delta p}_{IC}$ and $\mathbf{\Delta v}_{IC}$ are the initial position and velocity errors, respectively. In the case of attitude, as $\mathbf{\Psi}_{IC}^t$ is the true attitude, the following formulation was used:

$$\mathbf{\Psi}_{IC}^e = \mathbf{h}(\mathbf{F}(\mathbf{\Psi}_{IC}^t) \cdot \mathbf{F}(\mathbf{\Delta\Psi}_{IC})) \in \mathbb{R}^{3\times 1} \tag{12}$$

where $\mathbf{\Delta\Psi}_{IC}$ is the initial attitude error. Moreover, noise is added to the true position and attitude aiding measurements according to the following equations:

$$\mathbf{p}_m^e = \mathbf{p}_m^t + \mathbf{\Delta p}_{err} \in \mathbb{R}^{3\times 1} \tag{13a}$$
$$\mathbf{\Psi}_m^e = \mathbf{h}(\mathbf{F}(\mathbf{\Psi}_m^t)^T \cdot \mathbf{F}(\mathbf{\Delta\Psi}_{err})) \in \mathbb{R}^{3\times 1} \tag{13b}$$

where $\mathbf{p}_m^e, \mathbf{\Psi}_m^e$ are the erroneous, synthesized position and attitude measurements of the aiding sensor, respectively, $\mathbf{p}_m^t, \mathbf{\Psi}_m^t$ are the true position and attitude measurements, respectively, and $\mathbf{\Delta p}_{err}, \mathbf{\Delta\Psi}_{err}$ are the position and attitude errors added to the simulation, respectively.

As our policy model, we used a ResNet-based [33], end-to-end, fully convolutional network with 8 layers, all with 64 channels, dilated convolutions with a dilation factor of 2 and Relu activations. The input size is $H \times W$, and the output size is $H \times W \times 2$ —for the desired way-point and the next iteration way-point, respectively. In addition, we train using an ADAM optimizer [34] with a learning rate of 0.001. In each episode, we randomize the number of sand piles, shapes, locations, and volumes and the initial location of the bulldozer. In practice, the training dataset size for $agent_2$ includes more observations, as it includes multiple augmentations for each state. $agent_1$ converged after 100 episodes, while it took $agent_2$ 500 episodes to converge and reach the same grading performance.

Algorithm 1: Attitude interpolation $\mathbf{q}_n^1 \rightarrow \mathbf{q}_n^2$

Input: $\mathbf{q}_n^1, \mathbf{q}_n^2$ are two independent quaternions
Input: n_{int} is the number of interpolation points
Output: interpolated quaternions $\{\mathbf{q}_{interp}[k]\}_{k=0}^{n_{int}-1}$

$\otimes$ represents quaternion multiplication
$\mathbf{q} = [r, \mathbf{i}]^T \in \mathbb{R}^{4\times 1}$ represents the stacking of the real and imaginary parts to a quaternion.
$\mathbf{q}_{inc} = \frac{\mathbf{q}_n^1 \otimes \mathbf{q}_2^n}{\|\mathbf{q}_n^1 \otimes \mathbf{q}_2^n\|} \cdot \frac{1}{n_{int}}$ ▷ increment quaternion
for $k = 0$ **to** $(n_{int} - 1)$ **do**
  $\mathbf{q}_{inc}[k] = [q_{inc}^r, k \cdot \mathbf{q}_{inc}^{im}]$
  $\mathbf{q}_{inc}[k] = \frac{\mathbf{q}_{inc}[k]}{\|\mathbf{q}_{inc}[k]\|}$
  $\mathbf{q}_{interp}[k] = \frac{\mathbf{q}_n^1 \otimes \mathbf{q}_{inc}[k]}{\|\mathbf{q}_n^1 \otimes \mathbf{q}_{inc}[k]\|}$

Algorithm 2: IMU increment generation

Input: true trajectory samples $\{\mathbf{p}_m^n[k], \mathbf{\Psi}_m[k]\}_{k=0}^{K-1}$
Output: inertial increments $\{\mathbf{q}_t[k], \mathbf{q}_v[k]\}_{k=0}^{K-1}$

for $k = 0$ **to** $(K-1)$ **do**
- $\mathbf{v}^n[k] = (\mathbf{p}_m^n[k] - \mathbf{p}_m^n[k-1]) * \frac{1}{dt}$
- $\mathbf{q}_t[k] = \mathbf{h}(\mathbf{F}(\mathbf{\Psi}_m[k]) \cdot \mathbf{F}(\mathbf{\Psi}_m[k-1])^T)$
- $\mathbf{\Delta v}^b(k) = \mathbf{F}(\mathbf{\Psi}_m[k]) \cdot (\mathbf{v}^n[k] - \mathbf{v}^n[k-1])$
- $\mathbf{q}_v(k) = \mathbf{\Delta v}^b[k] - \mathbf{F}(\mathbf{\Psi}_m[k]) \cdot (0, 0, g \cdot dt)^T$

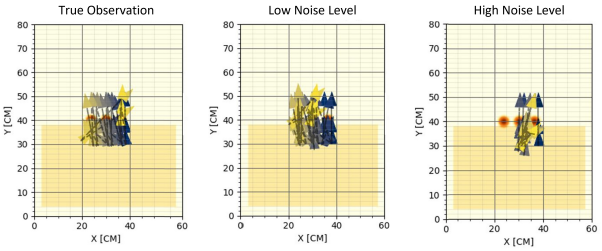
B. Simulation Results

1) Simulation setup: We first validate our claims (Hypotheses 1, 2) in a simulated environment, by running 50 rollouts of the same episode on the following scenarios:

- **Noise-Less:** In this setting, we generate clean, high-frequency inertial sensor measurements, as described in section IV-A, with no added noise. We then apply our navigation filter III-C algorithm in order to perfectly reconstruct the trajectory.
- **Sensor Fusion Noise:** In this setting, too, we generate a clean, high-frequency trajectory, as described for the *Noise-Less* setting, but we add IC and inertial sensor errors according to (12). The IC error values are: $\mathbf{\Delta p}_{IC} = (5, 5, 5)$ [cm], $\mathbf{\Delta v}_{IC} = (1, 1, 1)$ $[\frac{cm}{sec}]$, $\mathbf{\Delta \Psi}_{IC} = (4, 4, 5)$[deg]. In addition, the recorded (1σ) inherent measurement noise in the prototype environment setup was 1-2cm w.r.t position, and 1-2deg w.r.t orientation. During training, we added 2-3σ factors, e.g 5[cm] position and 5[deg] for orientation. From that reason, we set the aiding sensor noise to be $\mathbf{\Delta p}_{err} = \mathbf{\Delta}(x, y, z) = (5, 5, 5)$ [cm] for position and $\mathbf{\Delta \Psi}_{err} = \mathbf{\Delta}(\phi, \theta, \psi) = (1, 1, 5)$ [deg], according to (13). We then apply our sensor-fusion EKF algorithm (see section III-C) in order to reconstruct the trajectory, which now contains localization noise. This type of simulation allows us to mimic the uncertainty in a typical INS.
- **Extreme Noise:** This setting is similar to *Sensor Fusion Noise* but here the aiding sensor measurements contained a higher noise level, i.e., $\mathbf{\Delta p}_{err} = \mathbf{\Delta}(x, y, z) = (8, 8, 8)$ [cm] for position and $\mathbf{\Delta \Psi}_{err} = \mathbf{\Delta}(\phi, \theta, \psi) = (1, 1, 10)$ [deg]; all the steps from the previous section (see (13)) were followed under these conditions.
- **Increasing attitude noise:** We evaluated the effect of attitude noise, in particular $\Delta\psi$, where we compared the performance of $agent_1$ and $agent_2$ under increasing levels of measurement noise, i.e., $\mathbf{\Delta \Psi}_{err} = \mathbf{\Delta}(0, 0, \psi) = (0, 0, 0 \rightarrow 50)$ [deg]. Each point in Figure 5 is a mean of 50 grading episodes. The IC and other errors were the same as in section IV-B.1 above.

2) Simulation results analysis:

- **Noise-Less:** This simulation provides a baseline for comparison. As such, we consider actions taken under this setting as ideal, as can be seen in Figure 4a
- **Sensor Fusion Noise:** The results of $agent_1$ appear in Figure 4b, where it took the agent, on average over the 50 rollouts, 20% *more time* to complete the same task w.r.t. the setting without noise.
- **Extreme Noise:** The results of $agent_1$ appear in Figure 4c, where the agent did not complete the task, as some sand piles were left unattended.
- **Increasing attitude noise:** Figures 5a, 5b describe the behaviour of $agnet_1$. Notice that as the measurement noise level increases, both the total time to complete the episode and the total uncleared volume increase as well. This finding supports Hypothesis 1. Figures 5c and 5d describe the behaviour of $agnet_2$. Notice that though the noise level increases, the remaining uncleared volume decreases. We contend that this reduction highly correlates to the moderate increase in time spent on ***clearing sand*** in the case of $agent_2$. However, we argue that this is not the case for $agent_1$. In other words, $agent_2$ is more efficient in the presence of noise compared to $agent_1$, which supports Hypothesis 2.



(a) (b) (c)

Fig. 4. $agent_1$ simulation results with respect to the followed trajectory. The number of arrows represents the number of grading legs in an episode. Higher number suggests sub-optimality as it took the same agent *more time* to preform the *same task*. (**a**) Visualization of the followed trajectory without noise. (**b**) Visualization of the *same* scenario with low noise level. It is noticeable that the number of actions it took the agent to complete the task is higher in the presence of noise. (**c**) Visualization of the *same* scenario with high noise level. It is noticeable that the agent *did not* complete the task, as some of the sand piles are left untouched.

C. Scaled Prototype Results

In order to validate our Hypotheses 1, 2 under real-world conditions, we designed and built a $1:9$ scaled prototype environment (see Figure 6) that mimics several key aspects of a real-world environment. Our environment includes a RGB-D camera, mounted on top of a $250 \times 250cm$ sandbox and a scaled bulldozer prototype $60 \times 40cm$ in size. The RGB-D camera captures the entire scope of the target area and provides a realistic dense heightmap of it. In addition, it provides accurate pose measurements of the bulldozer by using an ArUco marker [32]. The bulldozer prototype itself is equipped with an inertial sensor VN-100 [35], which outputs the velocity and angular increments. These increments, which are processed by the INS algorithm, together with the pose output from the camera, are fed to the EKF filter, as described in Section III-C and illustrated in Figure 1. The heightmap images (observations) and fused poses from the EKF are then fed to the agent to predict its next actions. The agent then implements a low-level controller, moving according to the chosen destinations in a closed control loop manner. The three noise scenarios that were taken into consideration are:

1) **Noise-Less**: In this scenario, we used the measurements of the aiding system (with its inherent noise) as is, i.e., without additional noise.
2) **Sensor Fusion Noise**: Here, we added additional errors (other than the inherent system measurements noise) at

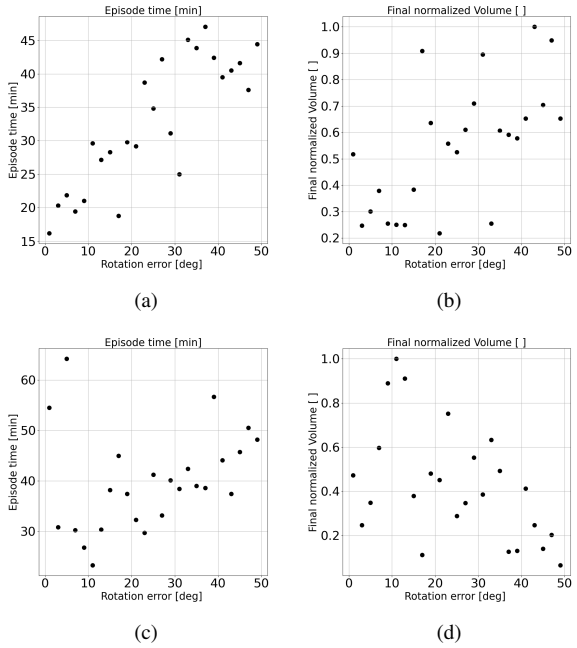


Fig. 5. Simulation of the performance of $agent_1, agent_2$ under increasing noise levels. (**a**-**b**) When noise level increases, so does the (**a**) time it takes $agent_1$ to complete the task and (**b**) the volume that remained uncleared. This suggests sub-optimal performance under uncertainties. (**c**-**d**) As noise level increases, the (**c**) time it took $agent_2$ to complete the task increases moderately, while the (**d**) total uncleared volume decreases. This suggests that $agent_2$ is more robust to uncertainties.

the level of $\mathbf{\Delta p}_{err} = \mathbf{\Delta}(x, y, z) = (5, 5, 5)$ [cm] for position and $\mathbf{\Delta \Psi}_{err} = \mathbf{\Delta}(\phi, \theta, \psi) = (0, 0, 5)$ [deg].

3) **Extreme Noise**: Here, too, additional errors (other than the inherent noise) were added, but at the level of $\mathbf{\Delta p}_{err} = \mathbf{\Delta}(x, y, z) = (5, 5, 5)$ [cm] for position and $\mathbf{\Delta \Psi}_{err} = \mathbf{\Delta}(\phi, \theta, \psi) = (0, 0, 10)$ [deg].

Fig. 6. Our lab experimental setup. Left - image of the measurement vision system located above the sand box and our bulldozer robot. Right - image of the robot as it approaches the sand pile in a grading episode.

Next, we describe the two experiments that were done on the real lab setup: (i) decision-making and (ii) full trajectory.

1) Decision-Making Under Uncertainty: In these experiments, we examine the quality of the decisions made by $agent_1$ and $agent_2$ under the scenarios described in IV-C.1.

Experiment setup:

In each scenario, 50 rollouts were collected; the results are shown in Table I. An action is classified as successful if over 50% of the blade capacity is filled with sand, i.e., sand will be spread in this leg. Figure 7 presents a visualization of successful and unsuccessful decisions.

Results analysis:

- **Noise-Less**: In this scenario, both agents took equally successful decisions, i.e $agent_1$ was a good selection for comparison. Moreover, though $agent_2$ was trained to cope with noise, its performance did not degrade due to the absence of noise (Table I, first row).
- **Sensor Fusion Noise**: In this scenario, $agent_2$ outperformed $agent_1$ by a large margin, making, on average, 40% better decisions (Table I, middle row). Here, the fusion algorithm was enabled for both agents, meaning that under residual noise, $agent_1$ made many unsuccessful decisions compared to $agent_2$. We used a simplified IMU error modeling (10) in simulation and training, while in practice an IMU has different types of errors, e.g. bias that increases with time, scale factor, misalignment etc. However, our agent was able to generalize to the real-world scenario. From this scenario we conclude that our training method can improve decision-making under uncertainties in real scenarios.
- **Extreme Noise**: In this scenario, both agents made some successful decisions and some unsuccessful decisions. Here, too, the performance of $agent_2$ was on par with that of $agent_1$ ($\approx 50\%$ for both; Table I, bottom row). This proves the first hypothesis, that noise reduces the performance of agents on the grading task. This extreme case aims to test out-of-distribution samples, and show the limitations of the training method. Quantifying this limit is a separate topic and proposed as future work. This regime is highly unlikely, as the model was trained to cope with 2-3σ factors of the inherent noise.

Accurate Decisions	$agent_1$	$agent_2$
Noise-Less	96%	98%
Sensor Fusion Noise	50%	90%
Extreme Noise	52%	56%

TABLE I. Percentage of successful decisions made by the agents on three scenarios (presented as the mean values of successful decisions over 50 roll-outs). Without noise, both agents exhibit the same level of performance. However, In the presence of noise, $agent_2$ made a successful decision in 90% of the cases, while $agent_1$ managed to do so only in 50% of them.

2) Full Trajectory Under Uncertainty: In these experiments, we aim to include all aspects of a real scenario, where not only the quality of the decisions are taken into account (as in Table I), but also the full cycle of perception, planning and path following in the presence of noise.

Experiment setup:

We measured the total time it took an agent to complete the episode on three scenarios discussed in IV-C.1. In each scenario, 3 rollouts were collected for both agents. Table II presents the mean time it took to complete the episode.

Results analysis:

- **Noise-Less:** Both agents took approximately 45 seconds to complete the task (Table II, first row).
- **Sensor Fusion Noise:** The performance of $agent_2$ degraded w.r.t the scenario without noise (105 sec rather than 45 sec), since residual noise still exists even though the fusion algorithm was enabled. $agent_1$ fared much worse: after making wrong decisions at the beginning of the episode, it followed the wrong path and did not even detect the next way-points in the episode.
- **Extreme Noise:** $agent_2$ completed the task after 156 seconds, highlighting the effect of noise on the path-

following algorithm. As $agent_2$ was trained to make better decisions under uncertainties, it took more time to finish the task due to errors in the control loop. Here, too, $agent_1$ diverged, for the same reasons as in the *Sensor Fusion Noise* scenario. Moreover, under *extreme noise* settings, $agent_2$'s performance degraded the most w.r.t the two prior scenarios (156 sec vs. 105, 45 sec).

From these experiments, we conclude that: (i) Though $agent_2$ made successful decisions in the *Extreme Noise* scenario, it still exhibits degraded performance compared to the *noise-less* scenario, since noise also affects the path-following algorithm. (ii) In the *Sensor Fusion Noise* and *Extreme Noise* scenarios, $agent_1$ diverged and did not complete the task due to unsuccessful decisions and did not manage to recover for the rest of the episode.

Scenario	$agent_1$[sec]	$agent_2$[sec]
Noise-Less	44	45
Sensor Fusion Noise	diverged	105
Extreme Noise	diverged	156

TABLE II. Results comparing two agents on three scenarios. Without noise, the performance of both agents was on par. However, In the presence of noise, $agent_2$ managed to complete the task in all three cases, while $agent_1$ diverged early on in the episode.

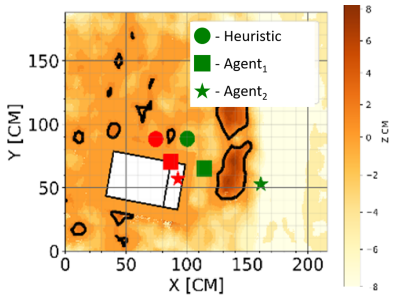


Fig. 7. Example of decisions in the form of way-points. Green markers represent the way-point an agent must reach in order to successfully grade sand in the current leg. Red markers represent the next leg's starting way-point the agent must reach. We compare the performance of three agents: (i) A heuristic agent presented in [18], represented by a round marker. (ii) $agent_1$ from [21], represented by a square marker. (iii) $agent_2$, represented by a star marker. Here, agents (i) and (ii) made sub-optimal decisions and will, therefore, not grade any sand during this leg. Agent (iii) made a successful decision and will, therefore, grade sand in this leg and successfully move to the next way-point.

V. Conclusions

In this work, we compare several grading policies under localization uncertainties. We observe that an agent trained with complete certainty of its pose will exhibit degraded performance when presented with real-world localization uncertainty at inference, as the agent is exposed to out-of-distribution observations. To cope with this issue, we devise a novel training regime, where the agent is presented with localization uncertainty during training. Through multiple evaluations, we show that the proposed training procedure *indeed* improves the performance compared to the baseline method by 40%. This improved performance is further validated via rigorous experiments, both in simulation and on a real-world scaled prototype. While the presented method is applied to the autonomous grading task, it can be applicable to any autonomous vehicle task. As future work, training $agent_2$ to operate under extreme noise could be considered, as the bound is currently set to these conditions.

References

[1] P. Pradhananga, M. ElZomor, and G. Santi Kasabdji, "Identifying the challenges to adopting robotics in the us construction industry," *Journal of Construction Engineering and Management*, vol. 147, no. 5, p. 05021003, 2021.

[2] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[3] G. K. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, "Registration of 3d point clouds and meshes: A survey from rigid to nonrigid," *IEEE transactions on visualization and computer graphics*, vol. 19, no. 7, pp. 1199–1217, 2012.

[4] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[5] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "Vinet: Visual-inertial odometry as a seq-to-seq learning problem," in *Proc. IEEE AAAI*, vol. 31, 2017.

[6] N. Yang, R. Wang, J. Stuckler, and D. Cremers, "Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry," in *Proc. ECCV*, 2018, pp. 817–833.

[7] D. Svensson and J. Sörstedt, "Ego lane estimation using vehicle observations and map information," in *2016 IEEE Intelligent Vehicles (IV)*, 2016, pp. 909–914.

[8] A. Noureldin, T. B. Karamat, and J. Georgy, *Fundamentals of inertial navigation, satellite-based positioning and their integration*. Springer, 2013.

[9] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, "Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization," in *Proc. IEEE ICRA*, vol. 2. IEEE, 1999, pp. 1656–1663.

[10] H. Taghavifar and A. Mardani, "Off-road vehicle dynamics," *Studies in Systems, Decision and Control*, vol. 70, p. 37, 2017.

[11] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proc. IEEE CVPR*, 2020, pp. 11 525–11 533.

[12] C. M. Romero, "Maneuver planning for highly automated vehicles," Ph.D. dissertation, Albert-Ludwigs-Universität Freiburg, 2021.

[13] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint:1812.03079*, 2018.

[14] A. R. Reece, "Introduction to terrain vehicle systems: M. g. bekker. univ. of michigan press, ann arbor." *Journal of Terramechanics*, vol. 7, pp. 75–77, 1970.

[15] P. A. Cundall and O. D. L. Strack, "A discrete numerical model for granular assemblies," *Géotechnique*, pp. 47–65, 1979.

[16] D. Sulsky, Z. Chen, and H. Schreyer, "A particle method for history-dependent materials," *CMAME*, vol. 118, no. 1, pp. 179–196, 1994.

[17] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8459–8468.

[18] C. Ross, Y. Miron, Y. Goldfracht, and D. Di Castro, "Agpnet–autonomous grading policy network," *arXiv preprint arXiv:2112.10877*, 2021.

[19] M. Hirayama, J. Guivant, J. Katupitiya, and M. Whitty, "Path planning for autonomous bulldozers," *Mechatronics*, 2019.

[20] R. Li, C. Zhou, Q. Dou, and B. Hu, "Complete coverage path planning and performance factor analysis for autonomous bulldozer," *Journal of Field Robotics*, vol. 39, no. 7, pp. 1014–1034, 2022.

[21] Y. Miron, C. Ross, Y. Goldfracht, C. Tessler, and D. Di Castro, "Towards autonomous grading in the real world," *arXiv preprint arXiv:2206.06091*, 2022.

[22] A. Stentz, J. Bares, S. Singh, and P. Rowe, "A robotic excavator for autonomous truck loading," *Autonomous Robots*, vol. 7, no. 2, pp. 175–186, 1999.

[23] I. Kurinov, G. Orzechowski, P. Hämäläinen, and A. Mikkola, "Automated excavator based on reinforcement learning and multibody system dynamics," *IEEE Access*, vol. 8, pp. 213 998–214 006, 2020.

[24] K. Oh, H. Kim, K. Ko, P. Kim, and K. Yi, "Integrated wheel loader simulation model for improving performance and energy flow," *Automation in Construction*, vol. 58, pp. 129–143, 2015.

[25] S. Khan, J. Guivant, and X. Li, "Design and experimental validation of a robust model predictive control for the optimal trajectory tracking of a small-scale autonomous bulldozer," *RAS*, vol. 147, p. 103903, 2022.

[26] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[27] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[28] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[29] Y. Miron and Y. Coscas, "S-flow gan," *arXiv preprint arXiv:1905.08474*, 2019.

[30] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[31] D. Ravat, "Analysis of the euler method and its applicability in environmental magnetic investigations," *JEEG*, vol. 1, no. 3, pp. 229–238, 1996.

[32] F. J. Romero-Ramirez, R. Munoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, 2018.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] https://www.vectornav.com/products/detail/vn-100, 2022, [Online].