Deep Multi-Modal Contact Estimation for Invariant Observer Design on Quadruped Robots

Anonymous Author(s)
Affiliation
Address
email

Abstract: This work reports on developing a deep learning-based contact estimator for legged robots that bypasses the need for physical contact sensors and takes multi-modal proprioceptive sensory data from joint encoders, kinematics, and an inertial measurement unit as input. Unlike vision-based state estimators, proprioceptive state estimators are agnostic to perceptually degraded situations such as dark or foggy scenes. For legged robots, reliable kinematics and contact data are necessary to develop a proprioceptive state estimator. While some robots are equipped with dedicated contact sensors or springs to detect contact, some robots do not have dedicated contact sensors, and the addition of such sensors is non-trivial without redesigning the hardware. The trained deep network can accurately estimate contacts on different terrains and robot gaits and is deployed along a contact-aided invariant extended Kalman filter to generate odometry trajectories. The filter performs comparably to a state-of-the-art visual SLAM system.

Keywords: State Estimation, Deep Learning, Quadruped Robot

1 Introduction

Legged robots can traverse uneven terrains. This unique capability gives legged robots the potential to conduct scientific exploration in extreme environments, execute rescue missions in hazardous scenes, and can help humans in everyday tasks [1]. To accomplish these goals, knowledge of the robot's current pose and velocity is necessary. This problem is known as *state estimation*. Methods for state estimation rely on visual and proprioceptive information. Vision-based methods take advantage of the rich information provided by cameras and LiDARs. However, these methods might suffer from failure due to illumination changes or extreme environments such as snowstorms or foggy scenes [2]. On the other hand, proprioceptive state estimators take only internal measurements and are thus agnostic to perceptually degraded situations. Moreover, unlike vision sensors that typically operate with low frequency (e.g., $10 - 60\,\mathrm{Hz}$), proprioceptive sensors usually operate at a high update rate (e.g., $100 - 2000\,\mathrm{Hz}$). This high-frequency information is crucial for controllers and planners to maintain stability and execute planned policies.

Proprioceptive state estimators often fuse the measurements from an Inertial Measurement Unit (IMU) with leg odometry. Leg odometry, in particular, uses kinematics and contact data to update the state. Reliable measurements of kinematic and contact information become critical in this case. Not all legged robots are equipped with dedicated contact sensors or springs to detect contact [3, 4]. The addition of dedicated contact sensors is non-trivial and often leads to a redesign of the hardware.

In this paper, we develop a deep learning-based contact estimator that does not require dedicated sensors and instead uses joint encoders, kinematics, and IMU data. We create contact data sets using an MIT Mini Cheetah robot [4] on eight different terrains. The trained network can correctly estimate contact events across different terrains and robot gaits. We further deploy the contact estimator along with a contact-aided invariant extended Kalman filter (InEKF) [5] and show that the resulting odometry trajectory is comparable to a state-of-the-art visual SLAM algorithm (used as a proxy for ground truth). The contributions of this work are as follows: 1) Open-source quadruped contact data sets recorded using an MIT Mini Cheetah, 2) A lightweight multi-modal deep learning-based contact estimator that mitigates the need for physical contact sensors and gives reliable estimations

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

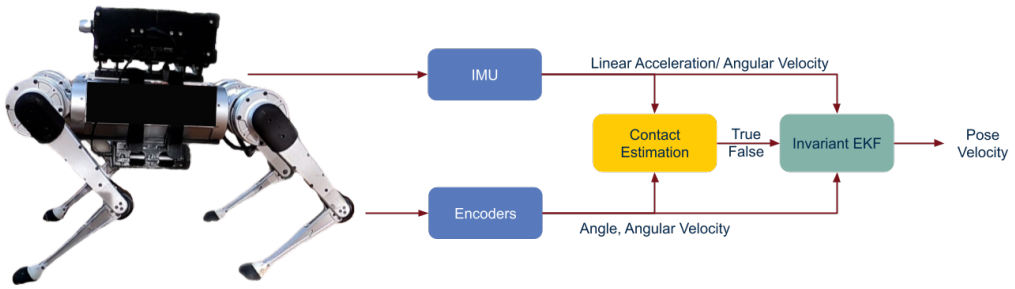


Figure 1: Proposed state estimator. The deep contact estimation network (Yellow) takes joint encoders data, IMU, and kinematics as input and classifies the correct contact state for the quadruped robot. The estimated contacts and the IMU and joint encoders data are fused inside a contact-aided invariant extended Kalman filter for state estimation. We tested the proposed pipeline on a MIT Mini Cheetah robot (Left).

on different terrains with different robot gaits, 3) A quadruped version of the contact-aided invariant EKF compatible with Lightweight Communications and Marshalling (LCM) [6, 7] interface. The data sets and software will be available for download after receiving the final decision, and 4) Experimental results for contact-aided state estimation using an MIT Mini Cheetah on eight different terrains.

2 Related Work

A thorough literature review on robot state estimation, including legged robots and contact-aided approaches, is available by Hartley et al. [5]. In this section, we focus on model-based and data-driven methods for contact estimation.

Model-Based Contact Estimation. Model-based approaches utilize physics models of the robot to estimate foot contacts. Traditionally, researchers segment the touchdown event of robot legs or prosthetic legs by thresholding on the estimated Ground Reaction Force (GRF) from the general equation of motion [8, 9, 10]. Although this method can detect touchdown events, the estimated GRF is often noisy and unreliable. De Luca et al. [11], Haddadin et al. [12] proposed a Generalized Momentum (GM) method for detecting contact events on robot manipulators. This GM-based method is, in fact, a filtered version of the work of Focchi et al. [8]. Although GM-based methods mitigate the noise problem in GRF estimation, an empirical threshold on the cut-off frequency is still required.

Hwangbo et al. [13] first introduces a probabilistic representation of the contact state and uses a method similar to a Hidden Markov Model (HMM) to fuse the dynamics and kinematics for contact estimation. They adopt a Monte-Carlo sampling algorithm to compute the transition model and verify their method against GM-based methods. Jenelten et al. [14] expands the HMM method and focuses on slippage detection. They demonstrate ANYmal [15], a quadruped robot, walking stably on slippery ground with their approach. The above two methods aim to detect contact as early as possible for the controller to maintain stability; however, we aim to detect reliable contacts for state estimation on various terrains. Since the mentioned works do not evaluate using a state estimator, it is unclear how their methods assist state estimation.

Bledt et al. [16] leverages both the GM-based methods and the probabilistic representation of contact states. They use a Kalman filter to fuse the gait phase scheduler information from the controller with the GRF estimated using the GM method. They demonstrate that this Kalman filtering approach can correctly estimate contacts and help the controller reduce the bouncing event upon touchdown. However, this method assumes the leg phase to be periodic as it uses the gait scheduler information in the prediction step of the Kalman filter. It could experience a loss in performance when the phase is heavily violated.

Data-Driven Contact Estimation. Data-driven approaches take advantage of the rapid development of recent machine learning techniques. Camurri et al. [17] uses logistic regression to learn the GRF threshold for contact detection. This work compares against heuristic-based thresholding on GRF using a base state estimator. The result shows that the logistic regression classifier can double the performance of the state estimator. However, compared to deep learning methods, the perfor-

mance of a logistic regression classifier gets saturated as the number of data increases [18, 19]. This method requires a specific training procedure for different gait, loading conditions of the robot, and individual terrain properties.

Rotella et al. [20] uses a fuzzy C-means clustering for the probability of contacts in all six end-effector degrees of freedom. They integrate the contact estimator with a base state estimator and show their approach performs considerably better than implementations that are purely based on measured normal force. However, this method assumes contact wrench sensors and additional IMU are available at each end-effector. Furthermore, this method was only tested in simulation. Its performance on real robots remains unknown. Piperakis et al. [21] proposes an unsupervised learning method for humanoid gait phase estimation. The authors employ Gaussian Mixture Models (GMMs) for clustering and show the accuracy by comparing to the ground-truth data and leg odometry. However, this work also assumes the availability of wrench/force sensors at each end-effector, and the clustering result is affected by the gait and data density.

The above methods either assume the availability of wrench/force sensors or are restricted by the nature of simple regression and are thus unable to generalize to different scenarios. In contrast, our work proposes a multi-modal deep learning-based contact estimator that does not require contact sensors and can generalize well to different gaits and terrain properties. Moreover, as more data becomes available, the network performance can be improved.

3 Background on Contact-Aided Invariant Extended Kalman Filtering

For a large class of systems defined on matrix Lie groups, the machinery of geometry provides natural coordinates that exploits symmetries of the space [22, 23, 24, 25]. The theory of invariant observer design is based on the estimation error being invariant under the action of a matrix Lie group. The fundamental result is that by correct parametrization of the error variable, a wide range of nonlinear problems can lead to (log) linear error equations [26, 27, 28]. We adopt the contact-aided invariant Kalman filtering approach [5] that is well-suited for legged robot state estimation problems.

State Representation. We wish to estimate the orientation, velocity, and position of the body (IMU) frame in the world frame, which are represented as $R_t \in \mathrm{SO}(3)$, $v_t \in \mathbb{R}^3$, and $p_t \in \mathbb{R}^3$, respectively. The subscript t here denotes the specific time stamp associated with the variables. We define the state $X_t \in \mathrm{SE}_{L+2}(3)$ as $X_t := (R_t, v_t, p_t, d_{lt})$, where $d_{lt} \in \mathbb{R}^3$ is the position of a contact foot in the world frame. When a contact event is detected for leg l, d_{lt} is calculated via the forward kinematics and the current body state. This d_{lt} is then augmented into the state to enforce the no-slip (zero-velocity) condition inside the filter. Similarly, if the contact constraint breaks, the associated contact position d_{lt} will be marginalized (removed) from the state to allow foot movement. The matrix Lie group, $\mathrm{SE}_{L+2}(3)$ is an extension of $\mathrm{SE}(3)$ and it was introduced by Barrau [27].

When a new contact event is detected, the position of the corresponding foot is computed using $\bar{d}_t = \bar{p}_t + \bar{R}_t h_p(\tilde{\alpha}_t)$, where $h_p(\tilde{\alpha}_t)$ is the foot position relative to the body frame computed by the forward kinematics, and the bar notation denotes estimated state variables. The state is represented dynamically so that we can utilize the contact information as an extra condition to regulate the estimation error. The number L indicates the number of legs that is currently having contact with the ground, and as we are working on a quadruped robot we will constrain this variable as $L \leq 4$.

Process Model. We assume the IMU measurements, i.e., angular velocity and linear acceleration, are corrupted by white Gaussian noise:

$$\tilde{\omega}_t = \omega_t + w_t^g, \quad w_t^g \sim \mathcal{GP}(0_{3,1}, \Sigma^g \delta(t-t')),\ \tilde{a}_t = a_t + w_t^g, \quad w_t^a \sim \mathcal{GP}(0_{3,1}, \Sigma^a \delta(t-t')),$$

where $\mathcal{GP}$ represents a Gaussian process and $\delta(t-t')$ is the Dirac delta function. Besides IMU measurements, we also assume the foot has zero velocity in the world frame during a contact event. However, to handle small foot slippage, we model the contact velocity noise in the body frame via a white Gaussian noise [29] $w_t^v \sim \mathcal{GP}(0_{3,1}, \Sigma^v \delta(t-t'))$. The process model of the individual term in the state becomes:

$$\frac{d}{dt}R_t = R_t(\tilde{\omega}_t - w_t^g)_\times,\ \frac{d}{dt}v_t = R_t\tilde{a}_t - w_t^a + g,\ \frac{d}{dt}p_t = v_t,\ \frac{d}{dt}d_t = R_t h_R(\tilde{\alpha}_t)(-w_t^v),$$

where $(\cdot)_\times$ denotes a 3×3 skew symmetric matrix, g is the gravity vector, $\tilde{\alpha}_t$ is the encoder measurements, and $h_R(\tilde{\alpha}_t)$ is the orientation of the contact frame in IMU (body) frame calculated by forward kinematics.

Measurement Model. In the filter update step, we use the contact constraint calculated earlier along with forward kinematics from current encoder measurements to correct the state. We assume the measurements from joint encoders are corrupted by white Gaussian noise: $\tilde{\alpha}_t = \alpha + w_t^g, \quad w_t^\alpha \sim \mathcal{N}(0_{3,1}, \Sigma^\alpha \delta(t-t'))$. The assumption that the contact point is fixed in the world frame during a contact event enables us to formulate the following measurement model using the position measurements of the forward kinematics as $h_p(\tilde{\alpha}_t) = R_t^\mathsf{T}(d_t - p_t) + J_p(\tilde{\alpha}_t) w_t^\alpha$, where $J_p(\tilde{\alpha}_t)$ is the analytical Jacobian of the forward kinematics function. Given this measurement model, we use the right-invariant correction model described by Hartley et al. [5] to update the state mean and covariance.

Implementation on Mini Cheetah. The implementation by Hartley et al. [5] is generalizable and open-sourced. In this work, we modify the open-sourced software for the MIT Mini Cheetah [4] by integrating the forward kinematics functions generated using FROST library [30]. An LCM interface is created for the filter to communicate with the robot. Furthermore, because the Mini Cheetah is not equipped with contact sensors, we develop a contact estimation framework to detect the contact events for the robot, as detailed in the next section.

4 Contact Estimation

In this section, we will discuss our deep learning approach for contact estimation. We define our contact state for each leg $l \in \{RF, LF, RH, LH\}$ as $C = [c_{RF} \quad c_{LF} \quad c_{RH} \quad c_{LH}]$ and $c_l \in \{0, 1\}$. Here, 0 denotes no contact, and 1 indicates a firm contact with the ground. Depending on the robot's motion, the contact state vector C can have totally 16 different states. (From all feet are in the air to all feet are in contact with the ground). We formulate it as a classification problem in our deep neural network and aim at estimating the correct contact state C given the input data. In order to follow typical classification pipelines, we map our contact state vector C to 16 different states $S \in \{0, 1, \ldots, 15\}$ by treating C as a binary value with 4 digits and using binary to decimal conversion as the function. For example, contact state $C_i = [0 \quad 1 \quad 1 \quad 0]$ is mapped to $S_i = 6$.

4.1 Preprocessing of the Input Data

The contact estimation network takes sensor measurements from joint encoders, IMU, and kinematics as input. For a synchronized time n, the sensor measurements are concatenated as a 54×1 array $z_n = [q_n \quad \dot{q}_n \quad a_n \quad \omega_n \quad p_{fn} \quad v_{fn}]$, where

$$
\begin{aligned}
q_n &= [q_{RF1n} \quad q_{RF2n} \quad q_{RF3n} \quad q_{LF1n} \quad \cdots \quad q_{LH3n}] \\
\dot{q}_n &= [\dot{q}_{RF1n} \quad \dot{q}_{RF2n} \quad \dot{q}_{RF3n} \quad \dot{q}_{LF1n} \quad \cdots \quad \dot{q}_{LH3n}] \\
a_n &= [a_{xn} \quad a_{yn} \quad a_{zn}]\,,\ \omega_n = [\omega_{xn} \quad \omega_{yn} \quad \omega_{zn}] \\
p_{fn} &= [p_{RFxn} \quad p_{RFyn} \quad p_{RFzn} \quad p_{LFxn} \quad \cdots \quad p_{LHzn}] \\
v_{fn} &= [v_{RFxn} \quad v_{RFyn} \quad v_{RFzn} \quad v_{LFxn} \quad \cdots \quad v_{LHzn}]\,.
\end{aligned}
$$

Here, q_n^T is a 12×1 vector containing all the joint encoder measurements (rad) at time n, $\dot{q}_n^\mathsf{T}$ is a 12×1 vector with joint angular velocity $(\mathrm{rad/sec})$, a_n holds the linear accelerations $(\mathrm{m/sec})$ from the IMU in the IMU frame, ω_n contains the angular velocity $(\mathrm{rad/sec})$ in the IMU frame, p_{fn}^T is a 12×1 vector with foot positions calculated from forward kinematics, and v_{fn}^T is a 12×1 vector that carries the linear velocities of each foot. It is worth noticing that both p_{fn} and v_{fn} are represented in the robot's hip frame. To infer the relationship between data across the time domain, for data point at time n, we create a window with size w and append previous measurements within this window into a 2D array $D_n = \left[z_{n-w}^\mathsf{T} \quad z_{n-w+1}^\mathsf{T} \quad \ldots \quad z_n^\mathsf{T}\right]^\mathsf{T}$. D_n is a $w \times 54$ array. (In our case, we use $w = 150$.) Each time, the network takes D_n as input and estimates the contact state S_n as output.

4.2 Network Architecture

The contact estimation network consists of 2 blocks of convolutions and 3 fully connected layers, as shown in Figure 2. Each block contains 2 1-dimensional convolution layers and a 1-dimensional max

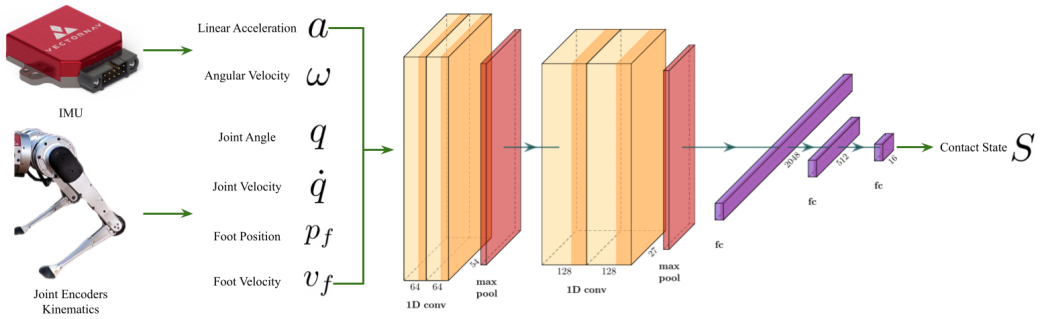


Figure 2: The architecture of the contact estimation network. The network takes in measurements from encoders and an IMU and outputs the contact state of the robot. The structure consists of 2 blocks and 3 fully connected layers. In each block, there are 2 1-dimensional convolution layers and 1 1-dimensional max pooling. The second convolution layer in each block is applied with a dropout mechanism to prevent overfitting.

Table 1: The proposed contact estimation network architecture. Conv1D stands for 1-dimensional convolution layer, Pool1D means 1-dimensional max pooling, and FC is the fully connected layer.

	Operator	Stride	Filters	Size	Output Shape
	Input				$N \times 54 \times w$
Block 1	Conv1D	1	64	3	$N \times 64 \times w$
	Conv1D	1	64	3	$N \times 64 \times w$
	Pool1D	2	64	2	$N \times 64 \times w/2$
Block 2	Conv1D	1	128	3	$N \times 128 \times w/2$
	Conv1D	1	128	3	$N \times 128 \times w/2$
	Pool1D	2	128	2	$N \times 128 \times w/4$
FC	FC1	-	-	-	2048
	FC2	-	-	-	512
	FC3	-	-	-	16

pooling. The convolution layers are designed to extract deep features from the input data. We choose a 1-dimensional kernel to increase computational efficiency in terms of memory usage and run time. The kernel moves along the time domain, and the data is padded to preserve dimensions. ReLU is employed as the nonlinear activation function for the convolution layer. The second convolution layer is applied with a dropout mechanism to prevent overfitting. At the end of each block, a 1-dimensional max-pooling layer is added to downsample the data.

The second block is connected to 3 fully connected layers that convert the deep features into the 16 classes we defined earlier. We also employ the dropout mechanisms in the first 2 fully connected layers to prevent overfitting. Finally, we formulate a classification problem using the cross entropy loss as $L(P_i) = -\log \frac{\exp(P_i)}{\sum_j \exp(P_j)}$. P_j is the probability output from the network of state j, and P_i is the probability of the ground truth state. The detailed network architecture is listed in Table 1.

4.3 Contact Data Set

We create open-source contact data sets using an MIT Mini Cheetah robot. We record all the proprioceptive sensor measurements from the robot as LCM logs. Recorded measurements include joint encoders data, foot positions and velocities, IMU measurements, and estimated joint torques from an MIT controller [4]. The IMU measurements are received at 1000 Hz, while other data are recorded at 500 Hz. All data are upsampled to match the IMU frequency after being recorded. Around 1,000,000 data points are collected from various terrains, including asphalt road, concrete, forest, grass, middle pebbles, small pebbles, rock road, and sidewalk. To give the network negative examples of non-contact scenarios, we also record several sequences of the robot walking in the air by holding the robot up and giving the same controller command. (i.e., not having contact with the ground while operating the same gait.)

The grass data sets are collected inside an outdoor facility equipped with a motion capture system, which we utilize as a proxy for ground truth trajectory. Markers are attached to each foot and the robot body for the motion capture system to record the absolute foot positions and body pose in the world frame. For the rest of the data sets, in addition to the proprioceptive sensors, we also record RGB-D images with an Intel D455 camera mounted on top of the robot. We use these images in a state-of-the-art visual SLAM system, ORB SLAM2 [31], to generate the approximate ground truth

Table 2: Number of data of each terrain in the contact data set.

				Terrain Type						
overall	air walking gait	air jumping gait	asphalt road	concrete	forest	grass	middle pebble	small pebble	rock road	sidewalk
1,013,441	44,386	48,972	94,615	465,144	72,144	103,392	44,442	52,669	45,819	58,115

(a) Robot Setup for data collection. (b) Different ground types in the contact data set.

Figure 3: (a) Robot configuration for all data sets except grass. Additional RGB-D images are recorded using an Intel D455 camera mounted on the robot. The images are used in ORB SLAM2 to generate ground truth trajectories. (b) Different ground types in the contact data set.

trajectories. Figure 3b shows the picture of each terrain. The concrete data sets are recorded in a lab environment with polished concrete. The forest data sets are collected in a forest with random leaves, woods, and plants scattered around the ground. The middle pebbles data sets are recorded on pebbles of size ranging from 3 to 7 cm. The small pebbles data sets are recorded on pebbles ranging from 0.5 to 2 cm. Rock road consists of dirt and rocks of random sizes.

As a supervisory signal for the network, the ground truth contact points are generated with the normal components of the foot positions (in the body frame). For the grass data sets, we use the positions from the motion capture system, while for other data sets, the foot positions from kinematics are used. The ground truth contact points are labeled by finding the local minima values of the signal in a post-processing manner. Although the process is primarily automated, we still manually inspect the generated ground truth contacts, and some manual labeling is required.

5 Experimental Results

We train and evaluate our contact network on the contact data sets discussed in the previous section. Some sequences are reserved for testing. The rest of the data are split into validation, testing, and training sets with the ratio of 15%, 15%, and 70%, respectively. TABLE 2 lists the number of data being used from each terrain. A window size of 150 is used to allow the network to infer from the time domain. Each data point is normalized along the time domain to prevent scaling issues. We shuffled the training data to ensure generalizability in the model and reduce overfitting. We set the batch size to 30 and use 10^{-4} as the learning rate. The training process took 1.5 hours on an NVIDIA GTX 3090 GPU for 30 epochs. We note that after about seven epochs, the network can converge.

5.1 Contact Estimation

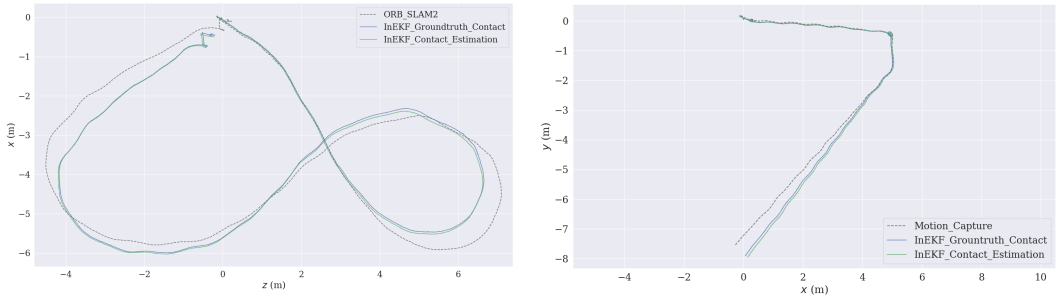
We now present the evaluation of the trained contact network. TABLE 3 lists the accuracy of the trained network on different terrains on the test sets. We evaluate the performance of the network in two ways. First, 16 class denotes the accuracy in terms of 16 contact states S we defined earlier. This case is a harsher way to evaluate the network since it requires all four legs to be correct simultaneously. Second, Leg RF to Leg LH list the accuracy of the individual leg by comparing the estimated contact state of each leg with ground truth contacts individually. We can see that, on average, the trained network achieves 87.54% of accuracy in terms of 16 contact states and 96.21% in each leg individually. Across different terrains, the network keeps an accuracy above 94% for each leg. Furthermore, the air walking and jumping data sets show that when the robot is walking with a similar kinematic pattern in the air, the network can tell there exist no contact event.

TABLE 3 shows the precision of the proposed network on different terrains. We use the function provided in scikit-learn [32] to compute the precision of the network. For each leg, the precision is defined as in typical binary classification problems. That is, $\frac{TP}{TP+FP}$, where TP is the number of

Table 3: Accuracy and precision of the proposed network on the test sets. 16 class denotes the accuracy and precision in terms of 16 contact states S. Leg RF to Leg LH list the accuracy and precision of individual leg by comparing the correctness of each leg individually.

Terrain	% Accuracy					
	16 class	Leg RF	Leg LF	Leg RH	Leg LH	Leg Avg
overall	87.54	96.73	95.69	95.94	96.50	96.21
air walking gait	100.00	100.00	100.00	100.00	100.00	100.00
air jumping gait	100.00	100.00	100.00	100.00	100.00	100.00
asphalt road	82.44	95.57	94.02	94.85	95.52	94.99
concrete	85.11	96.37	94.64	94.93	96.11	95.51
forest	87.22	95.59	97.04	96.24	96.13	96.25
grass	93.70	98.82	98.14	98.42	97.85	98.31
middle pebble	88.14	95.75	97.10	96.89	96.03	96.44
small pebble	85.06	95.59	94.63	96.32	96.07	95.65
rock road	88.01	98.32	96.57	95.38	96.31	96.65
sidewalk	89.11	96.79	95.95	96.38	96.44	96.39

Terrain	% Precision					
	16 class	Leg RF	Leg LF	Leg RH	Leg LH	Leg Avg
overall	86.52	94.46	92.67	93.48	93.65	93.57
air walking gait	N/A	N/A	N/A	N/A	N/A	N/A
air jumping gait	N/A	N/A	N/A	N/A	N/A	N/A
asphalt road	81.01	93.14	86.57	89.83	93.22	90.71
concrete	84.04	95.11	92.87	93.14	93.72	93.72
forest	85.75	89.96	94.86	95.07	92.72	93.11
grass	93.65	97.84	97.45	97.98	96.67	97.50
middle pebble	85.59	91.16	95.11	95.17	91.34	93.22
small pebble	84.37	90.21	90.56	94.27	92.34	91.90
rock road	85.76	98.21	94.07	91.74	93.45	94.43
sidewalk	87.93	94.69	92.19	96.21	93.98	94.24



(a) Concrete data set.

(b) Grass dataset.

Figure 4: (a) The trajectories of the concrete sequence. The estimated trajectory is mapped to the camera frame. (Y pointing downward, and Z pointing forward.) With the estimated contacts, the InEKF can reproduce a similar trajectory to that uses ground truth contacts. (b) The trajectories in the grass sequence. The trajectory is projected back to the world frame in the motion capture system. (Z pointed upward, and X forward.) With the estimated contacts, the InEKF can generate the trajectory similar to that uses ground truth contacts.

true positives and FP is the number of false positives. As for 16 classes, the precision is computed as a weighted average of the precision of each class by using the number of true instances for each class as the weight. Thus, precision demonstrates the ability to resist false positives, an important factor in practice. The proposed network maintains an average precision above 90% on all terrains.

5.2 Contact-Aided Invariant Extended Kalman Filter

The estimated contacts are used with the contact-aided invariant extended Kalman filter described in Section 3 on both grass and concrete test sets. The estimated contacts are converted into LCM messages along with IMU and encoder measurements. The InEKF takes in the LCM messages and estimates the pose of the robot accordingly. We also run the filter with the ground truth contact data to serve as a reference.

Figure 4a shows the trajectories of the concrete sequence. The blue line and green line are the InEKF trajectory with ground truth and estimated contacts, respectively. From the plot, we can see that the resulting trajectory with the estimated contacts has a small difference from the one with ground truth contacts. This indicates that the proposed contact estimator can accurately estimate the contact events for the state estimator. With the contact estimator, the InEKF can generate a comparable trajectory to a state-of-the-art visual SLAM system by only taking IMU and encoder measurements as input. We also compute the Relative Pose Error (RPE) of the estimated paths with the software developed by Grupp [33], as listed in TABLE 4. The RPE is calculated by finding the relative error of pose associated with each ground truth frame.

The trajectory of the grass sequence is presented in Figure 4b. It demonstrates that the proposed network can estimate contacts on different terrains. It also shows that the proposed contact estimator can correctly estimate contacts and reproduce a similar trajectory to the one that uses ground truth contacts. The RPE is also shown in TABLE 4. We notice that the RPE of the grass sequence is smaller than the concrete sequence. This trend is consistent with the accuracy shown in TABLE 3, and we conjecture it is because the polished concrete is slippery and the no-slip condition might be violated more often. The above experiments demonstrate that our contact estimation network enables reliable state estimation using only encoder and IMU measurements.

Table 4: The Root Mean Square Error (RMSE) of the Relative Pose Error in two different test sequences. The trajectories generated with estimated contacts and ground truth contacts achieves a almost identical RPE value. Unit: $(\mathrm{m}/frame)$.

	Concrete Sequence	Grass Sequence
Ground Truth Contacts	0.0416	0.0083
Estimated Contacts	0.0416	0.0083

Table 5: The average performance of different network structure on the test sets. Each "block" consists of 2 convolution layers followed by ReLU activation function and 1 max pooling, as described in Figure 2.

Network Structure	**% Accuracy**					
	16 class	Leg RF	Leg LF	Leg RH	Leg LH	Leg Avg
2 Blocks	87.54	96.73	95.69	95.94	96.50	96.21
1 Block	87.48	96.68	95.62	95.89	96.51	96.18
4 Blocks	85.87	95.61	95.54	94.83	95.39	95.34
Conv-Pool-Conv-Pool	86.45	96.16	95.36	94.90	96.00	95.60

5.3 Ablation Study and Runtime

We perform an ablation study on different network architectures to study how the performance changes. TABLE 5 lists the average performance of different network structure on the test sets. Each "block" consists of 2 convolution layers followed by ReLU activation function and 1 max pooling, as described in Figure 2. In the table, 2 Blocks is the proposed network detailed in Section 4. 1 Block refers to the network with only the first block and the fully connected layers of the proposed network. As for 4 Blocks, we added two additional blocks with 256 and 512 channels before the fully connected layers. Conv-Pool-Conv-Pool consists of only 2 convolution layers of sizes 64 and 128 and the fully connected layers. Each convolution layer is followed by a ReLU function and one max pooling. From the table, we can see that 2 Blocks has the best performance among all other structures. However, 1 Block maintains a smaller network while showing a slightly lower accuracy.

The inference speed on an NVIDIA GTX 3090 GPU is approximately 1100 Hz, and the inference speed on an NVIDIA Jetson AGX Xavier, which we equipped on the robot, is around 340 Hz. These results are without TensorRT optimization. The InEKF runs at 2000 Hz on an Intel i7-8750H CPU.

6 Discussion and Limitations

The developed contact estimator can accurately capture contact events for a quadruped robot without force/contact sensors. However, the contacts are modeled as binary values, and the “quality” of contacts is not modeled in our contact estimation network. As a result, a perfect contact is always assumed when a contact event is detected. Furthermore, the covariance matrix of the contact measurements is set heuristically in the InEKF, which along with the uncertainties in the kinematic models, could be the potential sources of estimation error in the InEKF. In the future, we wish to expand the contact detection network to have a covariance estimation of the current contact state. This approach will allow the state estimator to treat contact information as a sensor measurement with an online uncertainty update and potentially decrease the drift in the state estimation.

We formulated the problem as a classification problem with 16 different contact states. This model can potentially be further improved by employing a multi-task learning approach [34, 35]. This approach allows to independently segment the contact events for each foot instead of treating them as a whole. Currently, the contact estimation is done in an offline manner. More development is required to create an online interface between the PyTorch [36] contact estimator and the C++ state estimator.

7 Conclusion

We developed a multi-modal deep learning-based contact estimation method that does not require contact/force sensors and works well with different robot gaits on distinct terrains. We present open-source contact data sets that require minimal supervision for labeling contact events of an MIT Mini Cheetah robot. We show that the trained contact network can achieve 94% of accuracy across different terrains. The estimated contacts are employed in a contact-aided invariant state observer for quadruped robots, and the resulting trajectory is comparable to a modern visual SLAM system.

References

[1] M. H. Raibert, H. B. Brown Jr, M. Chepponis, J. Koechling, and J. K. Hodgins. Dynamically stable legged locomotion. Technical report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1989.

[2] M. Bijelic, T. Gruber, and W. Ritter. A benchmark for LIDAR sensors in fog: Is detection breaking down? In *IEEE Intelligent Vehicles Symposium*, pages 760–767. IEEE, 2018.

[3] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim. MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2245–2252. IEEE, 2018.

[4] B. Katz, J. Di Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6295–6301. IEEE, 2019.

[5] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle. Contact-aided invariant extended kalman filtering for robot state estimation. *International Journal of Robotics Research*, 39(4):402–430, 2020.

[6] A. S. Huang, E. Olson, and D. C. Moore. Lcm: Lightweight communications and marshalling. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4057–4062. IEEE, 2010.

[7] D. Moore, E. Olson, and A. Huang. Lightweight communications and marshalling for low-latency interprocess communication. 2009.

[8] M. Focchi, V. Barasuol, I. Havoutis, J. Buchli, C. Semini, and D. G. Caldwell. Local reflex generation for obstacle negotiation in quadrupedal locomotion. In *Nature-Inspired Mobile Robotics*, pages 443–450. World Scientific, 2013.

[9] S. A. Fakoorian, D. Simon, H. Richter, and V. Azimi. Ground reaction force estimation in prosthetic legs with an extended kalman filter. In *2016 Annual IEEE Systems Conference (SysCon)*, pages 1–6. IEEE, 2016.

[10] G. Fink and C. Semini. Proprioceptive sensor fusion for quadruped robot state estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 10914–10920. IEEE, 2020.

[11] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1623–1630. IEEE, 2006.

[12] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363. IEEE, 2008.

[13] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter. Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3872–3878. IEEE, 2016.

[14] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter. Dynamic locomotion on slippery ground. *IEEE Robotics and Automation Letters*, 4(4):4170–4176, 2019.

[15] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 38–44. IEEE, 2016.

[16] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim. Contact model fusion for event-based locomotion in unstructured terrains. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4399–4406. IEEE, 2018.

[17] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini. Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *IEEE Robotics and Automation Letters*, 2(2):1023–1030, 2017.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[20] N. Rotella, S. Schaal, and L. Righetti. Unsupervised contact learning for humanoid estimation and control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 411–417. IEEE, 2018.

[21] S. Piperakis, S. Timotheatos, and P. Trahanias. Unsupervised gait phase estimation for humanoid robot walking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 270–276. IEEE, 2019.

[22] G. S. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011.

[23] A. W. Long, K. C. Wolfe, M. J. Mashner, and G. S. Chirikjian. The banana distribution is Gaussian: A localization study with exponential coordinates. *Proceedings of the Robotics: Science and Systems Conference*, 265, 2013.

[24] T. D. Barfoot and P. T. Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693, 2014.

[25] T. D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017. doi:10.1017/9781316671528.

[26] S. Bonnabel, P. Martin, and P. Rouchon. Non-linear symmetry-preserving observers on Lie groups. *IEEE Transactions on Automatic Control*, 54(7):1709–1713, 2009.

[27] A. Barrau. *Non-linear state error based extended Kalman filters with applications to navigation*. PhD thesis, Mines Paristech, 2015.

[28] A. Barrau and S. Bonnabel. The invariant extended Kalman filter as a stable observer. *IEEE Transactions on Automatic Control*, 62(4):1797–1812, 2017.

[29] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. Remy, and R. Siegwart. State estimation for legged robots: Consistent fusion of leg kinematics and IMU. In *Proceedings of the Robotics: Science and Systems Conference*, 2012.

[30] A. Hereid and A. D. Ames. FROST: Fast robot optimization and simulation toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver, BC, Canada, 2017. IEEE.

[31] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[33] M. Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

[34] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. doi:10.1109/TKDE.2021.3070203.

[35] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.