Fast Simulation of Vehicles with Non-deformable Tracks

Martin Pecka[1,2], Karel Zimmermann[2], and Tomáš Svoboda[1,2]

***Abstract*— This paper presents a novel technique that allows for both computationally fast and sufficiently plausible simulation of vehicles with non-deformable tracks. The method is based on an effect we have called *Contact Surface Motion*. A comparison with several other methods for simulation of tracked vehicle dynamics is presented with the aim to evaluate methods that are available off-the-shelf or with minimum effort in general-purpose robotics simulators. The proposed method is implemented as a plugin for the open-source physics-based simulator Gazebo using the Open Dynamics Engine.**

I. INTRODUCTION

Tracked vehicles are often preferred over the wheeled ones in tasks where traversing complicated terrain is needed, such as in Urban Search and Rescue missions. Tracks provide higher stability, better traction and help the vehicle traverse holes in the underlying terrain.

It is common in robotics research that the initial development of algorithms is first conducted in a simulator to avoid excessive wear of the real vehicle. In some cases, an analytical description of the dynamics of the vehicle can be found, which may yield the best performance in predicting future states of the system. However, general-purpose simulators like Gazebo, V-REP, Webots, MORSE or Actin are often preferred, because setting up a simulation in these programs is easier than starting from scratch and they use approximate physics models that allow fast computation.

Simulation of wheels is straightforward in these simulators, thus all of them offer ways to simulate wheeled vehicles, including skid-steer motion for more proper steering of multi-wheel vehicles. However, for tracked vehicle simulation, there is no simple or straightforward approach, thus this kind of simulation is not available in most simulators. After an exhaustive search, only two simulators were found providing a tracked robot in its robot model library – the commercial simulators Webots and V-REP. However, none of these implementations is both plausible in difficult terrain and computationally light.

The most plausible and general simulation methods for rubber belts are based on finite elements analysis, where the belt is subdivided in many small elements that interact in a defined way. We omit this class of methods in this work due to their inherent excessive computational complexity, which makes them impractical for quick algorithm prototyping. Another argument for omitting these methods is that none of the most used open-source dynamics engines used in robotics supports simulation based on finite elements.

[1]Czech Technical University in Prague, Czech Institute of Informatics Robotics and Cybernetics

[2]Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics

In this paper, we present a novel technique for non-deformable tracks simulation, which we implemented in the open-source simulator Gazebo. The method provides a fast, simple and plausible simulation of tracks without grousers, and there is also a workaround for tracks with grousers. We compare this method with other already known simulation techniques. Finally, we propose a set of metrics that allow to compare the methods in terms of plausibility, computational time, and the range of track types that can be simulated by each of the respective techniques.

II. TYPES OF CATERPILLAR TRACKS

To clearly specify the type of vehicles this work is focused on, a short taxonomy of track types follows.

Based on the material the track is made of, the two basic types are *metal* tracks and *rubber* tracks. Metal tracks are usually made of many small *track plates* connected together with hinge-like joints. They are used wherever heavy load, high reliability and easy repairability are required. Rubber tracks are made of a continuous steel-reinforced band of rubber. The common use cases are lighter tracked vehicles (like robots) and conveyor belts.

Another distinctive feature of different track types is whether the outer shape of the track can be deformed. This is true for most metal tracks, but also deformable rubber tracks exist. The deformable track systems need a set of inner (sometimes also outer) wheels which keep the track in the required shape (see Figure 1). The track can bend in between the wheels, hence the name *deformable tracks*.

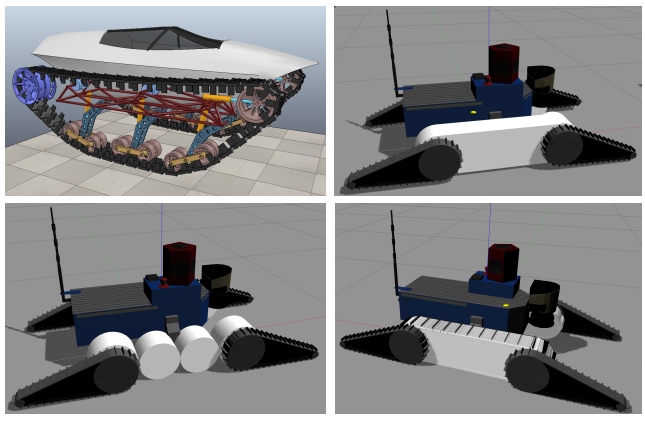

Fig. 1. **Track models.** ***Top left***: A vehicle with chain-like deformable tracks. This is the model available in model database of the V-REP simulator (courtesy of Qi Wang). ***Top right***: Non-deformable track model used for the proposed method. ***Bottom left***: Track approximated by 4 wheels. ***Bottom right***: Track made of 2 cm plates with grousers.

Non-deformable tracks have a solid guide (infill) in the sides of the track, which prevent the outer belt shape from being bent or deformed. The need for one or two driving wheels still remains (see Figure 1). This design is often chosen for rubber tracks, and it is the type this comparison focuses on.

A special category—*conveyor belts* and *escalators*—may be added to this taxonomy. These are used in industry as a continuous linear transportation mechanism. In many design principles they are similar to the tracks for vehicles, but the main difference is they are always fixed to the environment and thus have no dynamics as a whole.

Independently from the above categories, tracks can be equipped with *grousers*. These protrusions enlarge the contact surface and help increasing traction in soft materials. They are also depicted in Figure 1.

III. RELATED WORK

Depending on the purpose of the simulation, either very precise and detailed, or approximate models can be used. The former ones have been studied extensively in literature, whereas the approximate models, due to their triviality and inaccuracy, have not been examined that much despite their frequent use in nowadays robotics.

A. Precise Models

Simulation of the chain-like (metal) tracks can be completely set up using existing robotics simulators – they consist of a set of solid track plates connected with hinge joints, several wheels and, possibly, suspension of the wheels. All these components are available in simulators like Gazebo, Webots or V-REP. However, this type of simulations is both computationally intensive and very unstable for the high number of highly constrained dynamic elements [1]. Only the V-REP simulator provides a reliable simulation of this type, and it has to be very finely tuned to work. Sokolov et al.[2] tried to implement this method in Gazebo, but the reported results are quite unsatisfactory.

When the general-purpose simulators fail, specialized simulators were developed to simulate the chain-like track dynamics. Wallin et al. [3] compared several formulations of the mechanical joints when applied to metal tracks. They conclude that each formulation has its advantages and disadvantages and has to be chosen with respect to the specific use-case.

As discussed in the introduction, considerable effort is put into simulation of tracks using the Finite Elements Analysis [4], [5]. But the precision and computational demands are of higher orders than the methods we focus on.

In agriculture and military research, the track-soil interaction is of high interest (mainly due to sinkage of the track plates). Most of these works seem to only consider planar motion of the vehicle [6], [7], [8] and mainly concentrate on computing correct sinkage-induced behavior. Yamakawa and Watanabe [9] provide a fully three-dimensional simulation taking into account the track-soil interactions and wheel suspension.

B. Approximate Models

The models described in the previous section all have in common that they properly simulate some effects, but are either very computationally intensive, or neglect some other important effects (they e.g. assume only motion on flat ground with just small obstacles).

What the robotics community often needs is a fast model that can predict the approximate trajectory of a tracked robot given some control commands. The tolerance to the approximation error differs based on the environment and particular task to be simulated. We are not aware of any approximate model for the deformable track type, because its behavior is highly nonlinear and it essentially requires to model the individual parts of the track separately. The rest of this section thus concentrates on approximate models for non-deformable tracks, which generally also do not model soil sinkage.

In some environments, only flat ground is present and the robot is never intended to overcome an obstacle. This might be the case in household robotics or storehouse helper robots. Then there is effectively only a very small difference between a tracked robot and a 4-wheel robot with skid-steer control. In such scenarios, simulating the track by 4 wheels may be sufficient (with synchronized velocity of the wheels on each side).

In some cases, the tracks can be treated completely passive and the robot motion can be roughly estimated by setting zero friction to the track surface, and pushing the robot with a virtual force instead of driving the tracks. This force can be applied via a P(ID) controller, so that the robot achieves the desired velocity and keeps it. However, the usual effects of friction can not be simulated, so for example the robot can not stand on a tilted plane with no control force (which the real robot can).

When negotiation of obstacles needs to be accounted for, the 4-wheel approximation would fail, because the robot could not support itself on obstacle edges by the middle parts of the tracks. In this case, the problem is often solved by putting more virtual intersecting wheels inside the track. This approach has been tested by Sokolov et al. [10], and is available as a predefined model in V-REP and Webots simulators. The model still uses the skid-steer wheel control, has to properly synchronize wheel velocities on each side, and it still has problems imitating the skid-steer behavior properly. On the other hand, the robot is able to overcome some obstacles, can support itself by any part of the track, and the motion can be initiated in the "physically-correct" way by applying torque to the wheels. But the geometry of such model does not correspond to the real geometry, which is why these models cannot plausibly simulate e.g. climbing up a staircase. We have observed in Section V that this model also gets stuck in some cases where the real robot would continue going. These models also do not work very well with the standard *friction pyramid* approximation of friction direction – it is instead needed to use the more precise (and more computationally expensive) *friction cone* model.

In the V-REP simulator, we have found one more method of approximate simulation, but it is only suitable for conveyor belts and other static elements. It basically bypasses the physics computations by directly setting linear velocity of the whole conveyor belt mechanism, and resetting all forces acting on it in the next simulation step. This way, the conveyor belt can exert forces on objects colliding with it (because it has non-zero mass and velocity), but at the same time it stays on its place unaffected by any kind of dynamics (because the forces are zeroed-out each simulation step).

C. Skid-steer Motion

The slippage in the skid-steer behavior is an essential part of motion of tracked vehicles. While it automatically emerges from the precise simulation models as a result of track tension and other forces acting on the individual parts of the track, a kinematic model is also available for approximate or kinematic-only simulations.

Martínez et al. [11] define virtual points called *Instantaneous Centers of Rotation* (ICR), which depend on the desired turning radius and on a coefficient called *steering efficiency*. The robot follows a circular path centered at the ICR, and if the steering efficiency is equal to 1, the motion is the same as the motion of a geometrically equal differential-drive wheeled vehicle.

Janarthanan et al. [12] extend this theory for tracked vehicles with road wheels.

IV. Model Based on Contact Surface Motion

Our novel method exploits the dynamic simulation formulation as *Linear Complementarity Problem* (LCP), which is used in the Open Dynamics Engine (ODE) [13].

A. LCP Formulation

The dynamic simulation problem is an application of Newton's second law:

$$\mathbf{F} = Ma = \frac{d(M\dot{\mathbf{q}})}{dt} \tag{1}$$

where t is time, $\mathbf{F}$ is the force acting on the dynamic system, M is the mass and inertia matrix, and $\dot{\mathbf{q}}$ is the linear and angular velocity of the bodies (which is the derivative of the system state $\mathbf{q}$). The force $\mathbf{F}$ is split into *external force* $\mathbf{F_e}$ and *constraint force* $\mathbf{F_c}$ [1], which is a set of forces generated by joint constraints that keep joint constraints valid in the next time step.

The constraints are written in the form

$$\dot{C}(\mathbf{q}) = J\dot{\mathbf{q}} \geq 0 \tag{2}$$

where J is the *constraint Jacobian*. An observation in [1] states that the direction of the constraint force is given by J, so it is sufficient to search for the constraint force magnitude λ (so that $\mathbf{F_c} = J\,\lambda$).

In simulation, the derivative is discretized into short time steps Δt (usually 1 ms) and the state of the system is integrated step-by-step using Euler's integration [1]. The state of the system in the next time step $n+1$ can be expressed as

$$\mathbf{q_{n+1}} = \mathbf{q_n} + \mathbf{v_{n+1}}\Delta t$$

where the new velocity vector $\mathbf{v_{n+1}}$ (corresponding to $\dot{\mathbf{q}}$ in the continuous setting) is obtained from from Equation 1:

$$\begin{aligned}\mathbf{v_{n+1}} &= \mathbf{v_n} + M^{-1}(\mathbf{F_e} + \mathbf{F_c})\Delta t \\ &= \mathbf{v_n} + M^{-1}(\mathbf{F_e} + J\lambda)\Delta t\end{aligned}$$

The unknown constraint force magnitude λ is the solution of the following LCP [1]:

$$\begin{aligned} JM^{-1}J^T\lambda\Delta t + J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t) &\geq 0 \\ \text{given } \lambda \geq 0,\ J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t) &\geq 0 \\ (J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t))^T\lambda &= 0 \end{aligned}$$

B. Contact Constraint Equations

In each time step, when links L_1 and L_2 collide, a set of contact points $\{C_i\}_{i=0}^{N}$ is generated at places where the links touch or penetrate each other. Every contact point is assigned a *contact joint*, which is a temporary constraint between L_1 and L_2. The set of constraints yielded by the contact joint consists of a position constraint (repelling the two links from each other along the contact normal), and a velocity constraint for friction (stopping parallel motion of the two links), which often utilizes the Coulomb friction representation [14].

Linear velocity of L_1 is denoted by $\mathbf{v_1}$, angular velocity by $\omega_\mathbf{1}$, and $\mathbf{r_1}$ is the vector from the center of L_1 to C_i; respective definitions hold for L_2. Further, $\mathbf{t_i}$ denotes the main tangential friction direction (which is perpendicular to the contact normal).

The approximate velocity constraint for Coulomb friction at contact point C_i with friction coefficient μ_i is [14]:

$$\frac{\partial C_i}{\partial t} = (\mathbf{v_2} + \omega_2 \times \mathbf{r_2} - (\mathbf{v_1} + \omega_1 \times \mathbf{r_1})) \cdot \mathbf{t_i} = 0 \tag{3}$$

$$-\mu_i \leq \lambda_i \leq \mu_i \tag{4}$$

which can be interpreted as "stop any motion in direction $\mathbf{t_i}$". The LCP solver tries to find magnitude of the friction force in direction $-\mathbf{t_i}$ (which is bounded by μ_i) that would satisfy this equation.

C. Utilizing Contact Surface Motion

With the previous definitions, the novel method can be described as a modification of Equation 3. To account for the track velocity v_t, Equation 3 is adjusted to:

$$\frac{\partial C_i}{\partial t} = v_t$$

which can be interpreted as "find a force that would keep relative motion at velocity v_t". With this change, desired track velocities can be set and the model will start moving accordingly just by applying the modified friction constraints.

This model is however not able to correctly simulate grousers. If the real track has grousers, one way to add a similar effect to the simulation is to increase the friction coefficient. Despite it only looks like a workaround, this

method proved useful in our previous work [15] where we heavily utilized the simulator to find a control policy suitable also for the real robot.

D. Enabling Skid-Steer Motion

The last part to be defined is the friction direction t_i. If only forward motion is required, it can be simply set to be parallel to the tracks. This, however, causes problems when the model has to turn around using skid-steering motion (since the friction forces are not consistent with the turning manoeuver).

Here we connect the dynamic simulation with a kinematic model of tracked vehicle motion described in [11]. This model introduces a virtual point called *Instantaneous Center of Rotation* (ICR_v). This is a point lying at the common axis of both tracks in distance proportional to the turning radius (and influenced by track-soil interaction via a coefficient called *steering efficiency*). The whole vehicle is then said to be following a circular path centered at ICR_v (or driving straight if ICR_v is in infinity). Thus, we know the desired trajectory of all contact points on the track, and we set each direction t_i to be tangent to this trajectory, see Figure 2.

This model has been successfully used in our previous work [15]. Implementation of the proposed method (and some other) has been offered to the Gazebo community [16].

V. COMPARISON OF MODELS

In this section, a comparison of methods of modeling non-deformable tracks is presented.

A. Tested models

The tested models are described in the following sections (they are depicted in Figure 1). Each model is shortly described, and a shortcut for it is defined, which is used throughout the rest of the text and figures. All of the tested models differ only in representation of the main tracks – all other properties like mass, inertia, shape etc. were the same for all models.

With each of the models, identification of the most realistic set of parameters was done. The optimized parameters were always *linear* and *angular gain* – ratios that convert control inputs from simulator to velocity commands for the models. Other parameters were adding only for the models they make sense with, and consist of *steering efficiency* and friction coefficients in the first and second friction direction.

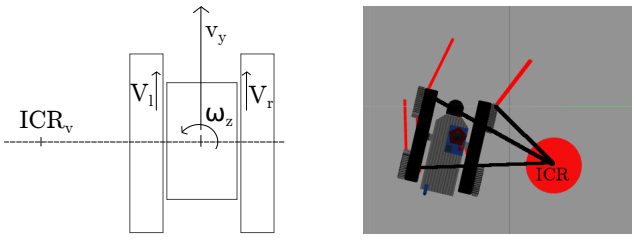

Fig. 2. **Instantaneous Center of Rotation.** ***Left***: A schematic view of the ICR_v. If the vehicle doesn't slip to the sides, ICR_v lies always on the depicted horizontal line passing through the centers of the tracks [11]. The distance of ICR_v from the center depends on forward velocity v_y and angular velocity ω_z (inverse kinematics), or the speeds of the left and right track V_l and V_r (forward kinematics). ***Right***: Computed direction of the friction forces (red lines) for the case where ICR_v lies in the center of the red disk. The friction forces are perpendicular to the (black) lines connecting the contact points with ICR_v.

First, we tried to manually find a suitable set of parameters and estimated the ranges for each of them. Then we did 5 iterations of optimization, in each of which we examined 5 samples from a multivariate Gaussian distribution centered on the so far best set of parameters (with covariance derived from the estimated ranges). Examination of each sample consisted of traversing all defined scenarios with model settings taken from the sample, and summing up the weighted metric values (defined further in this section). To account for the uncertainty in the simulator, each traversal was tried 3 times and the metric value was averaged over these trials.

1) Model based on Contact Surface Motion: This is the novel model shortened as **CSM**.

2) Wheels instead of tracks: Model with 4 wheels instead of each track (**4wheels**) or 8 wheels (**8wheels**). All wheels are velocity-controlled using a skid-steer wheel control mechanism (with wheels on each side synchronized in velocity).

3) Subdivision to plates: Model with belt subdivided into 10 cm plates (**plates10**) or 2 cm plates (**plates2**) interconnected by hinge joints, plus sprocket and idler wheels. Versions with grousers attached to the track plates are shortened as **plates10g** and **plates2g**. The inner space of the track is filled with a solid box which can collide with the track plates, thus emulating the non-deformability of the track. Only the sprocket wheel is controlled, using torque control. This model requires more tuning in the simulator. To simplify it, the sprocket wheel is represented by a cylinder with infinite friction with the track plates (so that it efficiently transfers force to them without the need to model the teeth and their interaction with the plates). Further, lateral motion of track plates has to be avoided (otherwise, they would slip off the track very easily). This would be best done with a planar joint, which is however not available in Gazebo/ODE. As a workaround, placing two virtual vertical plates to the sides of each track (that collide only with the track plates) yields a similar behavior (although it is not ideal).

4) No friction: Model with zero friction between the tracks and ground (**no_friction**). The collision shape of the track is the same as in the **CSM** model, but the friction of the track is set to zero, and the whole model is force-controlled by applying a virtual force at its center of mass. The applied force is always perpendicular to the vertical axis of the robot.

5) The real robot: The **real** robot was also part of the test. It is the Absolem platform used in Urban Search and Rescue project TRADR [17]. Position of the robot in 6D space was measured by an IMU combined with track and laser odometry [18].

B. Test Scenarios

The models were tested in the following scenarios. Each scenario specifies a different metric which shows how successful the model was, and was selected specifically to

TABLE I

NUMERICAL COMPARISON OF THE SIMULATION METHODS.

	Metric	**csm** (proposed)	**4wheels**	**8wheels**	**no_friction**	**plates10**	**plates2**	**plates10g**	**plates2g**
Straight	d_t	$\mathbf{0.1 \pm 0.0}$	$\mathbf{0.1 \pm 0.0}$	$\mathbf{0.1 \pm 0.0}$	0.2 ± 0.1	1.6 ± 0.0	1.3 ± 0.0	1.6 ± 0.1	0.5 ± 0.0
Rotate	d_ω	$\mathbf{0.1 \pm 0.0}$	0.5 ± 0.0	1.6 ± 0.0	$\mathbf{0.1 \pm 0.1}$	1.4 ± 0.6	1.0 ± 0.1	2.5 ± 0.2	3.1 ± 0.0
Circular	$\sum d_t$	157.8 ± 5.7	$\mathbf{45.5 \pm 1.7}$	116.9 ± 6.4	47.4 ± 2.2	210.5 ± 30.6	189.5 ± 4.7	564.9 ± 36.3	195.7 ± 6.5
Back&forth	d_{st}	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	1.3 ± 0.1	0.1 ± 0.0	2.6 ± 0.2	0.3 ± 0.0
Ramp	$\sum d_\omega$	$\mathbf{1.8 \pm 0.0}$	2.2 ± 0.1	$\mathbf{1.8 \pm 0.0}$	2.0 ± 0.0	4.6 ± 1.1	2.6 ± 0.4	10.5 ± 4.1	34.7 ± 1.9
	$\sum d_t$	8.0 ± 2.7	$\mathbf{4.6 \pm 1.3}$	5.3 ± 0.8	16.8 ± 4.3	36.3 ± 1.6	61.7 ± 0.2	36.9 ± 3.5	121.9 ± 1.1
Staircase	$\sum d_\omega$	14.0 ± 0.8	10.2 ± 0.1	10.7 ± 0.3	17.1 ± 0.4	36.0 ± 31.3	$\mathbf{8.4 \pm 1.8}$	25.1 ± 11.1	45.5 ± 1.8
	$\sum d_t$	$\mathbf{12.1 \pm 1.0}$	16.3 ± 2.3	15.2 ± 0.9	13.0 ± 3.7	111.5 ± 5.9	86.8 ± 2.5	14.4 ± 7.0	163.0 ± 1.6
Stand on st.	d_{st}	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	0.2 ± 0.0	0.2 ± 0.0	0.6 ± 0.6	0.2 ± 0.0	0.2 ± 0.2	0.2 ± 0.0
	d_ω	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	0.4 ± 0.4	0.1 ± 0.0	0.2 ± 0.2	0.1 ± 0.0
Pallet	$\sum d_\omega$	$\mathbf{27.6 \pm 0.9}$	27.7 ± 12.5	28.3 ± 17.7	31.8 ± 1.0	164.4 ± 151.5	47.6 ± 4.7	83.0 ± 13.1	64.3 ± 2.6
	$\sum d_t$	49.9 ± 2.5	116.9 ± 76.2	157.8 ± 46.6	$\mathbf{45.2 \pm 3.4}$	508.3 ± 146.3	181.1 ± 3.8	198.6 ± 72.7	78.4 ± 1.2
CPU time	time	38.9 ± 1.4	47.5 ± 1.7	82.4 ± 3.3	$\mathbf{33.0 \pm 1.3}$	254.9 ± 4.9	2282.6 ± 112.7	203.5 ± 8.3	2241.3 ± 31.8

Numerical results of the conducted experiments. Each model-scenario pair was executed 10 times, and the averages and standard deviations of the defined metrics are shown in the table. Shorthand d_t means the *distance to target point* metric (units are meters), d_{st} is distance from start. Term d_ω denotes smallest angular offset from target roll-pitch-yaw orientation (units are radians). Terms $\sum d_t$ and $\sum d_\omega$ stand for the *sum of positional errors* or *sum of angular errors* respectively (with units meters and radians). *CPU time* (in last row) is not a scenario, but as it is aggregated over all scenarios for each model, we display it as a row of values. The duration of all scenarios in simulation time is 110 seconds, so a run-time of 30 seconds means the simulation ran at $\frac{11}{3}$ realtime speed on the test notebook. Best results in each scenario are highlighted in bold for better orientation.

discover weak points of the models. All of the scenarios start with the robot in rest, no initial speed, forces or torques. A view on the obstacles in the scenarios is provided in Figure 3.

CPU time was measured in all scenarios. It represents the (real-world) time difference between the start of first scenario execution, and the end of the last scenario execution (so it is summed up over all scenarios for each model). The simulators were running with high process priority without an upper bound on performance.

Where a metric refers to the error from real robot trajectory, it means the scenario was traversed with the real robot, and the trajectory was recorded as a reference.

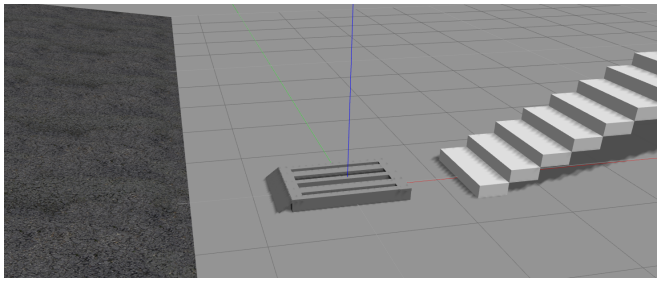

Fig. 3. **Obstacles used in test scenarios.** Obstacles that appear in the test scenarios (from the left): ramp, pallet, staircase. Also flat ground was used in scenarios. The models of the obstacles are 1:1 models of the obstacles traversed by the real robot.

1) Straight drive: Drive straight on a building floor using velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: distance from point $(3.0, 0.0, 0.0)^T$.

2) Rotating in place: Keep the center at one place while rotating at $0.6\ rad.s^{-1}$ for 10 seconds. Metric: Angular distance from heading $0.0\ rad$, metric distance from the starting point.

3) Circular path: Follow a circular path by driving left track at velocity $0.1\ m.s^{-1}$ and right track at velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz.

4) Ramp: Drive straight on a tilted ramp using velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

5) Staircase: Climb down a staircase using velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

6) Stand on staircase: Stand on a staircase with no control commands for 10 seconds. Metric: Distance from the starting point, angular offset from the starting orientation.

7) Pallet: Climb over a pallet using velocity $0.1\ m.s^{-1}$ for 30 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

8) Back and forth: Drive using velocity $0.2\ m.s^{-1}$ back and forth 10 times, with 2 seconds between every direction switch. Metric: distance from the starting point.

C. Test results

Each model was tested 10 times in each scenario, and the values of the metrics were averaged over these tests.

The detailed results are shown in Table I. A summary extracted from the test results is given in Table II.

It is evident from the table that the track plate models are slower by an order of magnitude or two than the other models, which might become a problem when there is need to execute many rollouts in the simulator (in [15], each learning episode took about 1 hour with *csm*; with *plates*, it would take several days). We have also observed, that the 10 cm plates are too rough approximation of the

TABLE II
SUMMARY RESULTS

	CSM	Wheels	Plates	No friction
Computation speed	✓	✓	×	✓
Plausibility on flat surfaces	✓	✓	✓	✓
Plausibility on rough terrain	✓	×	✓	×
Non-deformable tracks	✓	✓	✓	✓
Deformable tracks	×	×	✓	×
Grousers	×	×	✓	×

This table presents an overview based on the results of the conducted experiments. Sign "✓" means that the model is suitable for/supports the given use-case. Sign "×" means that the method is not suitable for/does not support the given use-case.

smoothly curved belt, and the resulting model's motion could be described as "bumpy". Last observation for track plate models is that without grousers, the robot is often not able to climb up the pallet (which, however, corresponds to the expected real behavior of a belt without grousers).

The wheeled models are computationally fast and provide good plausibility in most scenarios. They suffer from unrealistic slippage in the *stand on staircase* scenario, because the friction forces have unrealistic directions. The *pallet* scenario showed to be a big problem for these methods—if a sharp edge (e.g. a step or pallet edge) touches the track in a point where neighboring wheels intersect, the model suddenly stops moving as a result of unrealistic forces and their directions. We think it is not a bug in our implementation, since the same behavior was also observed with the wheeled track model available in V-REP simulator (which even uses a different dynamics engine—Bullet).

The *no_friction* model provided good results in all tested scenarios, except *stand on staircase*. That failure is obviously caused by the missing friction between tracks and ground. It was the fastest tested model.

The proposed *Contact Surface Motion* model was the second fastest tested model. It provided good results in all tested scenarios except *circular path*. Here, the parameter optimization was not able to find a set of parameters that would provide good performance for both *rotate in place* and *circular path*; with the best set of parameters, the robot was turning too quickly in the *circular path* scenario. Together with *no_friction*, only these two models traversed the pallet without problems.

VI. CONCLUSION AND FUTURE WORK

Simulation of tracked vehicles is a complicated task even when it is narrowed down to only simulation of non-deformable tracks. There is no method that would suit all needs that can arise in the robotics community, which might be the reason why only a minority of simulators support tracked vehicle models out of the box. The presented *Contact Surface Motion* is one of the fastest ones, but it provides highly plausible results in most cases. It can be easily used for non-symmetric tracks (along the up- or forward-axis).

The proposed set of metrics for comparison of the simulation models showed as a practical test for discovering weak and strong points of each model. Once the pull request to Gazebo [16] is merged, the testing world and obstacles [19] can be utilized by others to compare with their models.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union under grant agreement FP7-ICT-609763 TRADR; from the Czech Science Foundation under Project GA14-13876S, and by the Grant Agency of the CTU Prague under Project SGS16/161/OHK3/2T/13.

REFERENCES

[1] B. Kenwright and G. Morgan, "Practical Introduction to Rigid Body Linear Complementary Problem (LCP) Constraint Solvers," in *Algorithmic and Architectural Gaming Design*. IGI Global, 2012, pp. 159–201.

[2] M. Sokolov, I. Afanasyev, R. Lavrenov, A. Sagitov, L. Sabirova, and E. Magid, "Modelling a crawler-type UGV for urban search and rescue in Gazebo environment," in *Artificial Life and Robotics (ICAROB 2017), International Conference on*, 2017.

[3] M. Wallin, A. K. Aboubakr, P. Jayakumar, M. D. Letherwood, D. J. Gorsich, A. Hamed, and A. A. Shabana, "A comparative study of joint formulations: Application to multibody system tracked vehicles," *Nonlinear Dynamics*, vol. 74, no. 3, pp. 783–800, 2013.

[4] S. G. Arias, "Finite Element Analysis of Rubber Treads on Tracks to Simulate Wear Development," in *3DS Simulia Community Conference*, 2012.

[5] Z. D. Ma and N. C. Perkins, "A super-element of track-wheel-terrain interaction for dynamic simulation of tracked vehicles," *Multibody System Dynamics*, vol. 15, no. 4, pp. 347–368, 2006.

[6] G. Ferretti and R. Girelli, "Modelling and simulation of an agricultural tracked vehicle," *Journal of Terramechanics*, vol. 36, no. 3, pp. 139–158, 1999.

[7] B. Janarthanan, C. Padmanabhan, and C. Sujatha, "Longitudinal dynamics of a tracked vehicle: Simulation and experiment," *Journal of Terramechanics*, vol. 49, no. 2, pp. 63–72, 2012.

[8] D. Rubinstein and R. Hitron, "A detailed multi-body model for dynamic simulation of off-road tracked vehicles," *Journal of Terramechanics*, vol. 41, no. 2-3, pp. 163–173, 2004.

[9] J. Yamakawa and K. Watanabe, "A spatial motion analysis model of tracked vehicles with torsion bar type suspension," *Journal of Terramechanics*, vol. 41, no. 2-3, pp. 113–126, 2004.

[10] M. Sokolov, R. Lavrenov, A. Gabdullin, I. Afanasyev, and E. Magid, "3D modelling and simulation of a crawler robot in ROS/Gazebo," in *ICCMA '16, Proceedings of*, 2016.

[11] J. Martínez and A. Mandow, "Approximating kinematics for tracked mobile robots," *Journal of Robotics*, 2005.

[12] B. Janarthanan, C. Padmanabhan, and C. Sujatha, "Lateral dynamics of single unit skid-steered tracked vehicle," *International Journal of Automotive Technology*, vol. 12, no. 6, pp. 865–875, dec 2011.

[13] R. Smith *et al.*, "Open dynamics engine," 2005, http://www.ode.org/ode.html.

[14] J. Trinkle and J.-S. Pang, "On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction," *ZAMM - Journal of Applied Mathematics and Mechanics*, vol. 77, pp. 267–279, 1997.

[15] M. Pecka, K. Zimmermann, and T. Svoboda, "Autonomous Flipper Control with Safety Constraints," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. Los Alamitos, USA: IEEE, 2016, pp. 2889–2894.

[16] Gazebo pull request "Added support for tracked vehicles". [Online]. Available: https://bitbucket.org/osrf/gazebo/pull-requests/2652

[17] G. J. M. Kruijff et al., "Designing, developing, and deploying systems to support human-robot teams in disaster response," *Advanced Robotics*, vol. 28, no. 23, pp. 1547–1570, 2014.

[18] V. Kubelka and M. Reinstein, "Complementary filtering approach to orientation estimation using inertial sensors only," in *Proc. IEEE Int. Conf. Rob. Autom.*, 2012, pp. 599–605.

[19] Webpage containing all the data and configurations needed to replicate the experiments. [Online]. Available: http://cmp.felk.cvut.cz/~peckama2/simulation_quality/