2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Automatic Synthesis of Human Motion from Temporal Logic Specifications

Matthias Althoff, Matthias Mayer, and Robert Müller

Abstract— **Humans and robots are increasingly sharing their workspaces to benefit from the precision, endurance, and strength of machines and the universal capabilities of humans. Instead of performing time-consuming real experiments, computer simulations of humans could help to optimally orchestrate human and robotic tasks—either for setting up new production cells or by optimizing the motion planning of already installed robots. Especially when human-robot coexistence is optimized using machine learning, being able to synthesize a huge number of human motions is indispensable. However, no solution exists that automatically creates a range of human motions from a high-level specification of tasks. We propose a novel method that automatically generates human motions from linear temporal logic specifications and demonstrate our approach by numerical examples.**

I. Introduction

Properly setting up robots that share their workspace with humans is challenging since experiments involving humans are expensive and time-consuming. Consequently, one often wishes to synthesize realistic human motion from high-level specifications and optimize the setup and behavior of robots in computer simulations, e.g., to analyze the safety and throughput of the considered production cell. We review existing techniques for synthesizing human motion, identify research gaps, and summarize our contributions.

A. Related Work

There exists much work on synthesizing human motion from recombining motion capture data and physical simulations—both methods are often supported by machine learning techniques. However, there is only very little work on synthesizing motions from formally specifying high-level tasks, while there exists some work using non-formal specifications, such as annotations [1], verbs and adverbs [2], or even natural language [3], [4]. These works and other related works on human motion synthesis are reviewed below in the categories behavior planning, data-driven synthesis, simulation of physical models, and motion synthesis in industrial environments.

1) Behavior Planning: One of the simpler methods to create high-level behaviors is to use rule-based agents [5]. A more compact modeling formalism are behavior trees [6], which are used for robotic planning and character animation alike. Also, finite state machines are often used to create relevant behaviors [7]. When larger crowds of humans are simulated, one typically uses force models [8] or motion planning models with collision avoidance capabilities [9]. While crowd simulations typically focus on goal reaching, optimization-based methods also consider cost functions and constraints [10], and sometimes even annotations [1]. To explicitly consider probabilistic uncertainty in decision making, decision networks are used in [11] for high-level task generation. When acquisition of knowledge should be considered as well, cognitive models can be used [12]; these models have been combined with planning techniques in [13]. However, knowledge acquisition does not play a major role in repetitive tasks of human workers in factories. Yet another technique is to teach agents using programming-by-demonstration, where an agent learns a policy through observing experts [14].

2) Data-driven Synthesis: The aforementioned works perform the high-level behavior planning, but typically do not provide the exact movement of the human skeleton. One of the main reasons is that directly planning with all degrees of freedom in complex environments is computationally infeasible. Thus, one often uses motion capture techniques to adapt or learn typical movements of the human body.

Motion graphs are a popular way to combine segments of recorded motions through automatically-generated movements connecting closely-related motions [15]; an extension of this concept is presented in [16]. Besides recombining motion segments, one can also interpolate them, which is often referred to as motion blending [17] — techniques based on barycentric interpolation, radial basis functions, k-nearest neighbors, and inverse blending optimization are compared in [18]. By annotating motions with high-level characteristics, one can also interpolate between these [2].

Yet another possibility to synthesize motions from data is to complement sparse recordings. For instance, when only the position of the hands and feet are recorded, one can reconstruct likely poses of the entire body, which is also known as the inverse kinematics problem. Several learning-based approaches for this problem have been developed rather recently for variations of scaled Gaussian process latent variable models [19], [20] and Gaussian process dynamical models [21]; fully independent training conditional approximation has been used as well [22]. Inverse kinematics is combined with the aforementioned motion graphs in [23].

Inspired by the success in other domains like image processing and natural language processing, recurrent neural networks have been recently applied to motion synthesis. An autoencoder is used in [24] to learn a manifold of human motion so that unrealistic motions can be repaired; this work has been extended by stacking a deep feedforward neural

All authors are with the Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik und Echtzeitsysteme, Boltzmannstraße 3, 85748, Garching, Germany. {althoff,matthias.mayer,robert.mueller}@tum.de

network on top of the trained autoencoder [25]. For gait synthesis in rough terrain, a phase functional neural network has been proposed [26]. A variational generative model is learned in [27] to change the style of human motions. Some works can even synthesize human motion from natural language [3], [4].

3) Simulation of Physical Models: A disadvantage of data-driven methods is that the created movement might not be physically possible and that force-based interactions between humans themselves or between humans and the environment are not properly considered. An early work used physical models for animating human athletics [28]. Since then, physical models have gained interest in computer animations for the movie and game industry [29]. A challenge of physics-based animation is the computational complexity. Several works have addressed this challenge: By using objective functions and constraints that lead to linear time analytical first derivatives, computation times can be significantly reduced [30]. Another method is to find a low-dimensional space capturing the properties of the high-dimensional joint space of humans to run optimization algorithms on low-dimensional problems [31]. Yet another approach is to perform optimization in the joint space rather than using muscle activations by transforming constraints on muscle inputs to constraints in the joint space [32]. Parameters for realistic physical models are learned in [33] from motion-capture data.

Because it is difficult to develop the required control strategies for physics-based simulation so that characters can fulfill high-level tasks, machine learning methods are increasingly incorporated in physics-based simulation. In [34], a low-level controller for gait control and a high-level planner for step control is learned. Reinforcement learning is used in [35], [36] to find suitable control strategies.

4) Motion Synthesis in Industrial Environments: Most previous work for human motion synthesis has been performed for the movie and game industry, while only very few works exist on human motion synthesis in production settings. In [37], a 3D simulation for production environments with humans and robots is presented. Human motion synthesis for virtual maintenance is presented in [38], which is useful for maintainability design, supportability planning, and personnel training.

B. Contributions

We present the first work for synthesizing human motion given a rich temporal specification language. In particular, our contributions are:

- Synthesizing high-level task sequences from linear temporal logic.
- Creating variations of possible motions satisfying the temporal specification.
- We provide a modular framework so that motions can be generated from data-driven methods or physics-based simulation.
- Demonstrating the usefulness of our approach for a realistic production scenario.

This paper is organized as follows: Sec. II presents our high-level task specification and generation, from which we synthesize motions in Sec. III. We demonstrate the ability of our method for an industrial scenario in Sec. IV, followed by conclusions in Sec. V.

II. Specification and Synthesis of High-Level Tasks

We specify human tasks in production systems using linear temporal logic (LTL) [39] since it resembles natural language so that one can obtain LTL formulas from structured English [40].

A. Linear Temporal Logic

Since linear temporal logic is a formal specification language, there are no ambiguities compared to other forms of specification [41]. Given a sequence of states, let us introduce the operator $X\phi$ stating that ϕ holds in the next state and $\phi_1 U \phi_2$ stating that ϕ_1 holds until ϕ_2 holds. Given the set of atomic propositions AP, LTL formulas ϕ are constructed from atomic propositions $a \in AP$ by the following grammar [42, Def. 5.1]:

$$\phi ::= a | \neg\phi | \phi_1 \wedge \phi_2 | X\phi | \phi_1 U \phi_2. \tag{1}$$

Given a word σ and σ_i as the suffix of σ starting in the $(i+1)^{th}$ symbol, the semantics of LTL is defined inductively as [42, Fig. 5.2]:

$$\begin{aligned}
\sigma \models a &\iff a \in \sigma_0 \\
\sigma \models \neg\phi &\iff \sigma \not\models \phi \\
\sigma \models \phi_1 \wedge \phi_2 &\iff \sigma \models \phi_1 \text{ and } \sigma \models \phi_2 \\
\sigma \models X\phi &\iff \sigma_1 \models \phi \\
\sigma \models \phi_1 U \phi_2 &\iff \exists j \geq 0 \text{ s.t. } \sigma_j \models \phi_2 \text{ and} \\
&\qquad \forall\, 0 \leq i < j \text{ we have } \sigma_i \models \phi_1.
\end{aligned}$$

Other Boolean and temporal operator can be defined from the ones introduced above [42, Sec. 5.1]:

$$\begin{aligned}
\phi_1 \vee \phi_2 &= \neg(\neg\phi_1 \wedge \phi_2) \\
\phi_1 \implies \phi_2 &= \neg\phi_1 \vee \phi_2 \\
\phi_1 \iff \phi_2 &= (\phi_1 \implies \phi_2) \wedge (\phi_2 \implies \phi_1) \\
\text{true} &= \phi \vee \neg\phi \\
F\phi &= \text{true}\, U \phi \text{ (“in the future } \phi \text{ holds”)} \\
G\phi &= \neg F \neg\phi \text{ (“}\phi \text{ holds globally”)}
\end{aligned}$$

The unary operators bind stronger than the binary ones, $\neg$ and X bind equally strong, and the temporal operator U takes precedence over $\wedge$, $\vee$, and $\implies$.

B. Framework for Specifying Tasks of Human Workers

To guide the specification of human motion as much as possible, we provide specifications that always have to be considered and specifications that are specific to the given production environment. Thereto, we require several definitions: Given the continuous state $x \in \mathbb{R}^n$ of a human, we introduce the operator $occ(x) : \mathbb{R}^n \to P(\mathbb{R}^3)$ that returns the set of points occupied by the human, where $P()$ returns

the powerset of a set. When only the occupancy of a body part BP is required, we write $occ(x, BP)$. If the argument is an object O, the occupancy of that object is returned. Let us denote the space occupied by static and dynamic obstacles at time t as $\mathcal{O}(t)$. We define the predicate for collision as

$$\texttt{collision}(x(t)) \iff occ(x(t)) \cap \mathcal{O}(t) \neq \emptyset.$$

Let us further introduce the start region $\mathcal{R}_0 \subset \mathbb{R}^3$ as well as ξ intermediate regions $\mathcal{R}_i \subset \mathbb{R}^3$, which are regions the human worker has to touch. We will trigger predicates when entering certain regions, for which we define the predicate

$$\texttt{enter}(x(t), \mathcal{R}_i) \iff occ(x(t)) \cap \mathcal{R}_i \neq \emptyset.$$

When only a certain body part BP is concerned, we have

$$\texttt{enter}(x(t), BP, \mathcal{R}_i) \iff occ(x(t), BP) \cap \mathcal{R}_i \neq \emptyset.$$

If the first argument is an object, we define that

$$\texttt{enter}(O, \mathcal{R}_i) \iff occ(O) \cap \mathcal{R}_i \neq \emptyset.$$

When an object should be enclosed, we define

$$\texttt{enclose}(O, \mathcal{R}_i) \iff occ(O) \subseteq \mathcal{R}_i.$$

In this work, we consider the action primitives in Tab. I for an object O and regions $\mathcal{R}_i$, where regions can be arbitrarily small. Please note that we do not consider other actions[1], since they a) do not significantly change the occupancy of the human, b) can be easily abstracted by describing hand movements, or c) can be composed of other actions. To formalize the action primitives, we further require the ball $B(\epsilon) = \{x \in \mathbb{R}^3 | \, \|x\|_2 \leq \epsilon\}$ of a small radius ϵ and the Minkowski addition $\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$. The termination conditions in Tab. I are required to finish a task.

Once the predicates are evaluated for discrete points in time $t_k = k\,\Delta t$, where $k \in \mathbb{N}$ and $\Delta t \in \mathbb{R}^+$, we can consider them as atomic propositions of a word σ and remove the time dependency. The following specifications always have to be met:

- Initially, the human has to be part of the initial region:

$$\texttt{enter}(x, \mathcal{R}_0)$$

- In the future, the human should reach the final region:

$$F(\texttt{enter}(x, \mathcal{R}_\xi))$$

- The human should never collide with any obstacle:

$$G(\neg\texttt{collision}(x))$$

- Some actions can only be performed exclusively, such as `wait`.

Two examples of useful additional specifications are:

- After grasping object O, immediately carry it to region $\mathcal{R}_i$:

$$G(\texttt{grasp}(O) \implies X(\texttt{carry}(O, \mathcal{R}_i))$$

- At least once, after moving the right hand to $\mathcal{R}_i$, either insert object O_1 in O_2 or move object O_1 to $\mathcal{R}_j$:

$$F\big(\texttt{rightHand}(\mathcal{R}_i) \implies X(\texttt{insert}(O_1, O_2) \vee \texttt{move}(O_1, \mathcal{R}_j))\big)$$

Next, we show how to create task sequences from LTL specifications.

C. Synthesis of Task Sequences

A naive way to construct task sequences of human workers from LTL specifications is to randomly create task sequences and rejecting sequences that do not fulfill the specification. While checking LTL formulas is only linear in the length of the word σ [43], the number of possible task sequences is exponential in the length of the sequence.

To guide the creation of valid task sequences, we synthesize an automaton that fulfills an LTL formula. This problem, however, is double exponential in the length of the given specification [44]. Since most of the specifications we require are short, we did not suffer from this worst-case complexity in practice. If one is just interested in a single behavior and the specification can be formulated using the GR(1) fragment of LTL, the computational complexity is only polynomial with respect to the size of the state space [45]; most specifications can be formulated using the GR(1) fragment of LTL [45], [46].

Possible task sequences, however, are infinite since LTL specifies infinite traces. There exist synthesis approaches for LTL specifications of finite traces [47]. However, the synthesis of finite traces is not computationally cheaper [47] so that we simply traverse the synthesized automaton for the LTL specification in a way that terminal conditions are rather quickly reached using coverage algorithms for testing of systems [48].

III. Synthesizing Motions from Tasks

Given an ordered sequence of tasks S, such as $S = (\texttt{walk}(\mathcal{R}_1), \texttt{grasp}(O_1), \texttt{move}(O_1, \mathcal{R}_2), \texttt{release}(O_1), \texttt{sit}(chair), \texttt{grasp}(O_2), \texttt{carry}(O_2, \mathcal{R}_3))$, it remains to synthesize an appropriate motion. Most motion synthesis algorithms surveyed in Sec. I use motion-capture data. To focus on the novel aspects of synthesizing motions from temporal logic, we use standard approaches to recombine and adapt motion capture data as shown in [15].

A. Motion Automaton

We represent humans using marker positions selected according to the CMU Mocap database[2]: a human pose is given by the vector $p \in \mathbb{R}^{3m}$ consisting of the x,y,z-coordinates of each of the m markers.

Definition 1 (Motion Primitive): We define a motion primitive as a tuple $mp = (M, B, ta)$, where

[1]Examples of actions for which one of the subsequent points a)-c) holds are: a) turn head, foot motion (without leg motion), hand motion (without arm motion), screw, drill, turn object, hold object, apply pressure; b) cut, pick object, welding, grinding, bend object; c) tool preparation, tool positioning, tool use, attach object, detach object, latch object, unlatch object, check object, align object, remove object, assemble object, open object, and close object.

[2]`http://mocap.cs.cmu.edu/`

TABLE I

TASKS WHILE NOT WALKING, EXCEPT FOR WALK AND CARRY (O: OBJECT, LH: LEFT HAND, RH: RIGHT HAND, LK: LEFT KNEE, RK: RIGHT KNEE).

Name	Description	Termination condition
$\texttt{wait}(t)$	do not move much for duration t	$\forall \delta \in [0,t] : x(t^* + \delta) \in x(t^*) \oplus B(\epsilon)$; t^* is the time at which the task starts
$\texttt{move}(O, \mathcal{R}_i)$	move O to region $\mathcal{R}_i$	$\texttt{enclose}(O, \mathcal{R}_i)$
$\texttt{leftHand}(\mathcal{R}_i)$	move left hand inside $\mathcal{R}_i$	$\texttt{enclose}(x, lh, \mathcal{R}_i)$
$\texttt{rightHand}(\mathcal{R}_i)$	move right hand inside $\mathcal{R}_i$	$\texttt{enclose}(x, rh, \mathcal{R}_i)$
$\texttt{insert}(O_1, O_2)$	insert O_1 into O_2.	$\texttt{enclose}(O_1, O_2)$
$\texttt{extract}(O_1, O_2)$	extract O_1 from O_2.	$\neg\texttt{enter}(O_1, O_2)$
$\texttt{grasp}(O)$	grasp O.	$\texttt{enter}(x, rh, occ(O) \oplus B(\epsilon)) \vee \texttt{enter}(x, lh, occ(O) \oplus B(\epsilon))$
$\texttt{release}(O)$	release O.	$\neg\texttt{grasp}(O)$
$\texttt{sit}(O)$	sit on object O.	$\texttt{enter}(x, buttocks, occ(sittingSurface(O)) \oplus B(\epsilon))$
standUp	stand up.	$\neg\texttt{enter}(x, head, occ(ground) \oplus B(chestHeight))$
bend	bend (no bending of knees).	$\texttt{enter}(x, head, occ(ground) \oplus B(chestHeight)) \wedge$ $\neg\texttt{enter}(x, buttocks, occ(ground) \oplus B(kneeHeight))$
stoop	stoop (bending of knees).	$\texttt{enter}(x, head, occ(ground) \oplus B(chestHeight)) \wedge$ $\texttt{enter}(x, buttocks, occ(ground) \oplus B(kneeHeight))$
kneel	kneel.	$\texttt{enter}(x, lk, occ(ground) \oplus B(\epsilon)) \vee \texttt{enter}(x, rk, occ(ground) \oplus B(\epsilon))$
$\texttt{walk}(\mathcal{R}_i)$	walk to region $\mathcal{R}_i$.	$\texttt{enter}(x(t), \mathcal{R}_i)$
$\texttt{carry}(O, \mathcal{R}_i)$	hold O while walking to $\mathcal{R}_i$.	$\texttt{enter}(x, \mathcal{R}_i) \wedge \texttt{enclose}(O, \mathcal{R}_i)$

- $M \in \mathbb{R}^{3m \times \nu}$ is a motion concatenated by ν human poses $p \in \mathbb{R}^{3m}$ relative to a root marker (here: hip marker).
- $B \in \mathbb{R}^{6 \times \nu}$ is the sequence of positions (x, y, z coordinates) and orientations (around x, y, z coordinates) of the root marker over time t_k, $k < \nu$.
- ta is a task from Tab. I associated with the motion primitive. □

We distinguish between *gait primitives* ($ta \in \{\texttt{walk}, \texttt{carry}\}$) and all other primitives, which we refer to as *goal primitives*. While the subsequent theory applies to gait and goal primitives, we limit the search in Sec. III-B to subproblems only containing gaits to prune the search space.

Motion automata (aka *maneuver automata* [49] or *motion graphs* [15]) are used to formalize which motion primitives can be connected with each other; slight modifications between similar concepts exist. In this work, the states of the motion automaton refer to the motion primitives and the transitions between states model the connectivity between them. Since motions are represented by states of the automaton, we do not define a separate motion alphabet as in [49].

Definition 2 (Motion Automaton): In this work, a motion automaton is a tuple $MA = \{\mathcal{MP}, \Delta, \mathcal{MP}_0\}$, where

- $\mathcal{MP}$ is a finite set of motion primitives mp_i, which are the discrete states of the motion automaton;
- Δ is the set of discrete transitions $\Delta \subseteq \mathcal{MP} \times \mathcal{MP}$. A transition from mp_i to mp_j is denoted by (mp_i, mp_j);
- $\mathcal{MP}_0 \subset \mathcal{MP}$ is a set of possible initial motions. □

The semantics of the motion automaton is described informally. Starting from a motion primitive $mp_0 \in \mathcal{MP}_0$, the possible successive motions are $\{mp_i | (mp_0, mp_i) \in \Delta\}$, which in turn have a set of successive motions.

We compute the set of transitions Δ from a set of motion primitives $\mathcal{MP}$ as in [15] by an average distance over k_{comp} time steps. After introducing M_i^l as the l^{th} pose of the i^{th} motion, we calculate the average distance between the motion at the end of mp_i (having ν_i poses) and the motion at the beginning of mp_j. Since our motion primitives are invariant with respect to position and orientation of the root marker within the floor plane, we apply a homogeneous transformation using identical rotation matrices for each marker around the y-coordinate represented by $T \in \mathbb{R}^{3m \times 3m}$ and the translation vector $\lambda \in \mathbb{R}^{3m}$ shifting the x-coordinates and z-coordinates of all markers identically:

$$d_{ij} = \min_{T,\lambda} \frac{1}{k_{comp}} \sum_{k \in [1, k_{comp}]} ||M_i^{\nu_i - k_{comp} + k} - (T\, M_j^k + \lambda)||_2.$$

We add edges (mp_i, mp_j) to Δ if $d_{ij} < \delta$, where δ is a user-defined threshold. For simplicity, we use a global threshold δ in this work, however, one can adjust this for different types of movement, of course.

B. Motion Planning

For each task S_i in the task sequence S, we have to concatenate motion primitives according to the desired actions and the motion automaton. We use A* search to expand collision-free gait primitives in between tasks. Let us introduce ta_j as the task of mp_j. After each expansion with a gait primitive mp_q, we check whether a goal primitive $mp_g \in \{mp_j | (mp_q, mp_j) \in \Delta \wedge ta_j = S_i\}$ for the next task S_i would reach the next intermediate region $\mathcal{R}_i$.

The duration in between tasks $t_{i+1} - t_i$ is chosen as the cost function for A* and the minimum remaining time to the goal is chosen as the heuristic function $h(\psi) = \texttt{dist}(occ(\psi), \mathcal{R}_i)/v_{max}$, where ψ is a node of the A* search, $occ(\psi)$ returns the occupancy in node ψ, and $\texttt{dist}()$ returns the minimum Euclidean distance between two objects.

In practice, we found that choosing the absolute maximum velocities led to a huge increase in nodes expanded. Therefore, we relaxed the heuristic and chose lower velocities, which no longer guarantee optimal paths. We consider this as

an acceptable compromise since the cost function of humans is rather unknown [50].

C. Motion Blending

Since transitions between motion primitives are not always smooth, we blend motions between concatenated motion primitives [15]. We use a third-order polynomial $w(t)$ to blend between consecutive motions M_i and M_j over a specific number of frames nf. We ensure C^1 continuity of $w(t)$ by setting its coefficients such that $w(t_0) = 1$, $w(t_{nf}) = 0$ and $\dot{w}(t_0) = \dot{w}(t_{nf}) = 0$. The blended motion M_{syn} for $k \in [0, nf]$ is obtained as:

$$M_{syn}^{\nu_i - nf + k} = w(t_k) M_i^{\nu_i - nf + k} + (1 - w(t_k)) M_j^k. \quad (2)$$

Analogously, we blend the positions and orientations of the root marker.

IV. Numerical Examples

We demonstrate our approach using motion primitives from the CMU Mocap database. To obtain gait primitives, we split walking motions every 20 frames (corresponding to 166.6 ms) and goal primitives were manually generated by specifying start times t_{start} and end times t_{end}. Transitions were added to the motion automaton if their distances were below $\delta = 0.135$. Due to limited space, we intentionally limit our numerical examples to two specifications for an assembly task using the shorthand $\texttt{assemble}(\mathcal{R}_i) = \texttt{leftHand}(\mathcal{R}_i) \wedge \texttt{rightHand}(\mathcal{R}_i)$.

Specification 1 (S_1: Assembling one part): The human should repeatedly grasp a part O_1 followed by assembling it on the workbench in region $\mathcal{R}_1$. □

Specification 2 (S_2: Assembling two parts): The human should repeatedly grasp two parts O_1 and O_2 followed by assembling them on the workbench in region $\mathcal{R}_1$. □

The object O_1 is located at $[-1, 0.2, 1.0]^T$, O_2 at $[-1.5, 0.3, -1.2]^T$ and the region $\mathcal{R}_1$ is specified as the three-dimensional interval $\mathcal{R}_1 = [0.9, 1.1]^3$. The LTL formula for specification S_1 requires the atomic propositions $g = \texttt{grasp}(\texttt{O}_1)$ and $a = \texttt{assemble}(\mathcal{R}_1)$:

$$g \wedge G(a \implies F(g) \wedge g \implies F(a)) \quad (3)$$

As mentioned in Sec. II-B, we do not have to explicitly formulate the specifications that always have to hold, such as collision avoidance and exclusive actions, i.e., $G(\neg(a \wedge g))$. The specification of S_2 can be formulated similarly.

A. Creating and Analyzing the Synthesized Motions

From the specification we synthesize automata with *SPOT*[3] whose possible executions return valid task sequences; the automaton for S_1 is shown in Fig. 1. Its main loop between states 3 and 4 lets the human grasp O_1 and assemble it over and over again; the self-transitions in states 2 and 3 can be used to realize other tasks. Snapshots from the generated motions solving S_2 are shown in Fig. 2 and the matched goal primitives are listed in Tab. II.

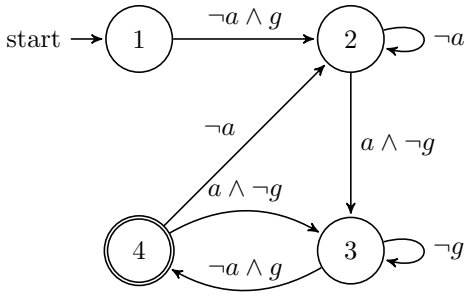

Fig. 1. Büchi automaton solving S_1. Since we only require finite sequences, we only ensure that the acceptance condition is passed at least once. The absorbing state which is reached when violating the specification is not shown for brevity.

TABLE II
MATCHED GOAL PRIMITIVES.

Task	**Goal region center: $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ in m**	**Matched goal primitive: Clip nr. ($\mathbf{t_{start}} \rightarrow \mathbf{t_{end}}$)**
Grasp	$[-1, 0.2, 1.0]$	69_73 (250 → 530)
Grasp	$[-1.5, 0.3, -1.2]$	64_26 (174 → 495)
Assemble	$[1.0, 1.0, 1.0]$	62_06 (80 → 2480)
Grasp	$[-1.5, 0.3, -1.2]$	69_73 (250 → 530)
Grasp	$[-1, 0.2, 1.0]$	69_75 (228 → 474)
Assemble	$[1.0, 1.0, 1.0]$	62_06 (80 → 2480)

As a first indicator for the quality of our synthesized motion, we present the global position of the right hand of the human in Fig. 3 for both specifications. The lack of jumps indicates smooth and natural movements. Also, the motion shows no overshoots so that the synthesized motion reaches the goal region of the next task without major detours; this indicates that the adapted heuristic returns almost optimal paths. The second plot shows that there is variation in the solution to S_2, as either O_1 or O_2 may be picked up first for assembly.

All computations have been performed in Python 3 on a laptop with a 9^{th} generation Intel i7 processor (2.6 GHz). The computation times for both specifications are listed in Tab. III. Currently, most of the computation time is spent on motion planning and creating the human skeleton. We expect that better heuristics and an improved implementation can significantly reduce these times.

TABLE III
COMPUTATION TIMES.

Specification	$\mathbf{S_1}$	$\mathbf{S_2}$
Synthesize Büchi automaton [s]	< 0.1	< 0.1
Create task sequence [s]	1.0	1.1
Motion planning [s]	614.2	283.9
Create motion of human skeleton [s]	39.4	47.9
Total computation time [s]	654.7	333.0

[3] https://spot.lrde.epita.fr/

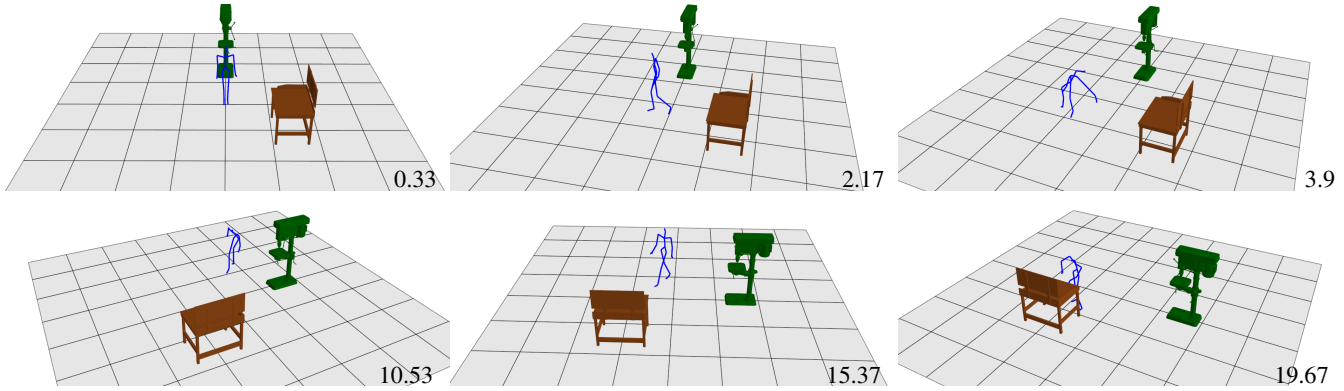

Fig. 2. The synthesized motion for S_2 at 0.033, 2.17, 3.9, 10.53, 15.37, and 19.67 seconds in a simple factory environment consisting of a green drill press and a brown workbench. The screenshot at 0.033 [s] shows the avatar at its initial position. At 2.17 [s] it walks to O_1, next O_1 and O_2 are grasped at 3.9 [s] and 10.53 [s], respectively, at 15.37 [s] the avatar walks to the workbench, where at time 19.67 [s] its hands are assembling the object. The full video is available at `https://mediatum.ub.tum.de/1536881?show_id=1554326`.

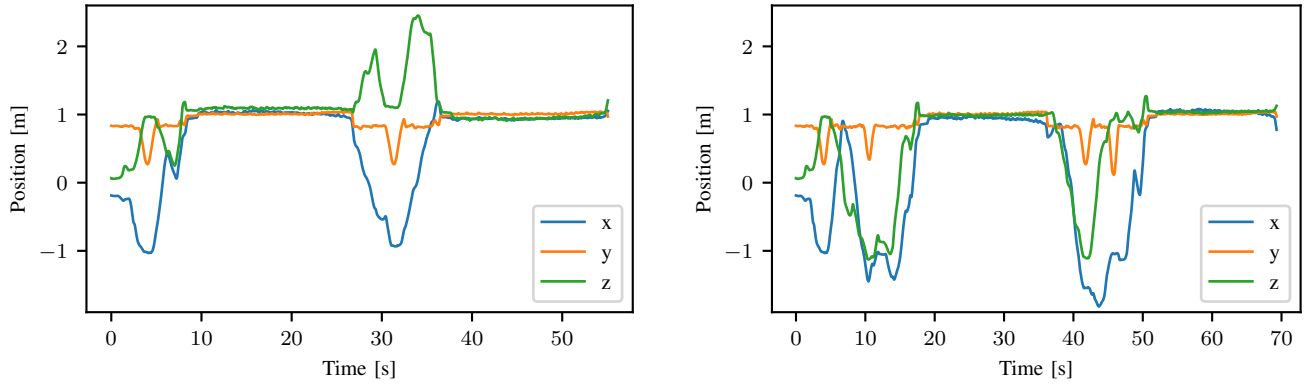

Fig. 3. Right hand position (x, y, and z coordinates) over time complying with S_1 (left plot) and S_2 (right plot). Objects are picked up from the floor every time the y position approaches a value close to zero. Longer stretches of small movements are where the assembly takes place.

V. Conclusions

We have presented the first approach that synthesized human motion in production environments from linear temporal logic specifications. Since linear temporal logic is close to natural language and many specification patterns for linear temporal logic exists, users can generate a variety of motions without having to tediously modify recorded motion primitives. We plan to use the generated motions for analyzing and optimizing human-machine coexistence. Other possible uses are planning of production facilities, machine learning of robots for human-machine co-existence, and estimation of cycle times for production cells.

Acknowledgements

We gratefully acknowledge partial financial support by the European Research Council (ERC) project justITSELF under grant agreement No 817629 as part of the EU Horizon 2020 program.

References

[1] O. Arikan, D. A. Forsyth, and J. F. O'Brien, "Motion synthesis from annotations," in *Proc. of the International Conference on Computer Graphics and Interactive Techniques*, 2003, pp. 402–408.

[2] C. Rose, M. F. Cohen, and B. Bodenheimer, "Verbs and adverbs: Multidimensional motion interpolation," *IEEE Computer Graphics and Applications*, no. 5, pp. 32–40, 1998.

[3] M. Plappert, C. Mandery, and T. Asfour, "Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks," *Robotics and Autonomous Systems*, vol. 109, pp. 13–26, 2018.

[4] H. Ahn, T. Ha, Y. Choi, H. Yoo, and S. Oh, "Text2Action: Generative adversarial synthesis from language to action," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2018, pp. 5915–5920.

[5] K. Perlin and A. Goldberg, "Improv: A system for scripting interactive actors in virtual worlds," in *Proc. of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996, pp. 205–216.

[6] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a unified behavior trees framework for robot control," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2014, pp. 5420–5427.

[7] M. Lau and J. J. Kuffner, "Behavior planning for character animation," in *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005, pp. 271–280.

[8] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, pp. 487–490, 2000.

[9] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environments," in *Proc. of the Symposium on Interactive 3D Graphics and Games*, 2008, pp. 139–147.

[10] M. Kapadia, S. Singh, G. Reinman, and P. Faloutsos, "A behavior-authoring framework for multiactor simulations," *IEEE Computer Graphics and Applications*, vol. 31, no. 6, pp. 45–55, 2011.

[11] Q. Yu and D. Terzopoulos, "A decision network framework for

the behavioral animation of virtual humans," in *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2007, pp. 119–128.

[12] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive modeling: Knowledge, reasoning and planning for intelligent characters," in *Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 29–38.

[13] M. Xu, Z. Pan, M. Zhang, P. Lv, P. Zhu, Y. Ye, and W. Song, "Character behavior planning and visual simulation in virtual 3D space," *IEEE MultiMedia*, vol. 20, no. 1, pp. 49–59, 2013.

[14] J. Dinerstein, P. K. Egbert, and D. Ventura, "Learning policies for embodied virtual agents through demonstration," in *Proc. of the 20th International Joint Conference on Artifical Intelligence*, 2007, pp. 1257–1262.

[15] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 473–482, 2002.

[16] J. Min and J. Chai, "Motion graphs++: a compact generative model for semantic motion analysis and synthesis," *ACM Transactions on Graphics*, vol. 31, no. 6, 2012, article 153.

[17] L. Kovar and M. Gleicher, "Flexible automatic motion blending with registration curves," in *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, pp. 214–224.

[18] A. Feng, Y. Huang, M. Kallmann, and A. Shapiro, "An analysis of motion blending techniques," in *Proc. of the International Conference on Motion in Games*, 2012, pp. 232–243.

[19] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *ACM transactions on graphics*, vol. 23, no. 3, 2004, pp. 522–531.

[20] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun, "Continuous character control with low-dimensional embeddings," *ACM Transactions on Graphics*, vol. 31, no. 4, 2012, article 28.

[21] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[22] X. Wu, M. Tournier, and L. Reveret, "Natural character posing from a large motion database," *IEEE Computer Graphics and Applications*, vol. 31, no. 3, pp. 69–77, 2011.

[23] M. Mahmudi and M. Kallmann, "Analyzing locomotion synthesis with feature-based motion graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 5, pp. 774–786, 2013.

[24] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," in *SIGGRAPH Asia Technical Briefs*, 2015, article 18.

[25] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM Transactions on Graphics*, vol. 35, no. 4, 2016, article 138.

[26] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017, article 42.

[27] H. Du, E. Herrmann, J. Sprenger, K. Fischer, and P. Slusallek, "Stylistic locomotion modeling and synthesis using variational generative models," in *Motion, Interaction and Games*, 2019, article 32.

[28] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien, "Animating human athletics," in *Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995, pp. 71–78.

[29] T. Geijtenbeek and N. Pronost, "Interactive character animation using simulated physics: A state-of-the-art review," in *Computer Graphics Forum*, vol. 31, no. 8, 2012, pp. 2492–2515.

[30] A. C. Fang and N. S. Pollard, "Efficient synthesis of physically valid human motion," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 417–426, 2003.

[31] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 514–521, 2004.

[32] Y. Jiang, T. Van Wouwe, F. De Groote, and C. K. Liu, "Synthesis of biologically realistic human motion using joint torque actuation," *ACM Transactions on Graphics*, vol. 38, no. 4, 2019, article 72.

[33] C. K. Liu, A. Hertzmann, and Z. Popoviundefined, "Learning physics-based motion style with nonlinear inverse optimization," in *Proc. of the International Conference on Computer Graphics and Interactive Techniques*, 2005, pp. 1071–1081.

[34] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[35] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," 2017, arXiv:1707.02286.

[36] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, 2018, article 143.

[37] A. Antakli, E. Hermann, I. Zinnikus, H. Du, and K. Fischer, "Intelligent distributed human motion simulation in human-robot collaboration environments," in *Proc. of the 18th International Conference on Intelligent Virtual Agents*, 2018, pp. 319–324.

[38] J. Geng, X. Peng, B. Qiu, Q. Wu, C. Lv, Z. Wang, and D. Zhou, "Simulation data integration-based approach for motion synthesis in virtual maintenance," *The International Journal of Advanced Manufacturing Technology*, vol. 99, no. 5, pp. 1481–1501, 2018.

[39] A. Pnueli, "The temporal logic of programs," in *Proc. of the 18th Annual Symposium on Foundations of Computer Science*, 1977, pp. 46–57.

[40] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Translating structured english to robot controllers," *Advanced Robotics*, vol. 22, pp. 1343–1359.

[41] N. G. Leveson, "Intent specifications: An approach to building human-centered specifications," *IEEE Transactions on Software Engineering*, vol. 26, no. 1, pp. 15–35, 2000.

[42] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[43] K. Havelund and G. Roşu, "Synthesizing monitors for safety properties," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2002, pp. 342–356.

[44] A. Pnueli and R. Rosner, "On the synthesis of a reactive module," in *Proc. of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1989, pp. 179–190.

[45] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Proc. of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation*, 2006, pp. 364–380.

[46] S. Maoz and J. O. Ringert, "GR(1) synthesis for LTL specification patterns," in *Proc. of the 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 96–106.

[47] G. De Giacomo and M. Y. Vardi, "Synthesis for LTL and LDL on finite traces," in *Proc. of the 24th International Conference on Artificial Intelligence*, 2015, pp. 1558–1564.

[48] S. Mouchawrab, L. C. Briand, Y. Labiche, and M. Di Penta, "Assessing, comparing, and combining state machine-based testing and structural testing: A series of experiments," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 161–187, 2011.

[49] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[50] S. Albrecht, K. Ramírez-Amaro, F. Ruiz-Ugalde, D. Weikersdorfer, M. Leibold, M. Ulbrich, and M. Beetz, "Imitating human reaching motions using physically inspired optimization principles," in *Proc. of the 11th IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 602–607.