# Overcoming the Limits of a Neural Network for Character-Scene Interactions

Marco Mameli(✉), Dario De Carolis, Emanuele Frontoni, and Primo Zingaretti

Dipartimento di Ingegneria dell'Informazione (DII), Università Politecnica delle Marche, Via Brecce Bianche, 60131 Ancona, Italy
m.mameli@pm.univpm.it,{e.frontoni,p.zingaretti}@univpm.it

**Abstract.** The motor synthesis of humanoid characters is one of the main problems in data-driven animations, with applications in robotics, entertainment and game development. Both in the commercial and academic fields there is a strong interest in developing new synthesis techniques. The attention of researchers in this field has recently turned to artificial intelligence techniques with the use of neural networks, in particular recurrent neural networks (RNN) that are well suited to predict data sequences, such as animations. The use of RNNs for the generation of animations despite the high success in the scientific field and the excellent results in real development, still has limitations that prevent its use on a larger scale. In this work, therefore, the need to overcome these limits has required to focus on the phase following the generation of interactions by the network. In particular, starting from a work present in the literature, the limitations were analyzed and solutions were proposed that made it possible to improve the visual rendering of the Carry and Sit operations. The results obtained are positive and did not require any intervention on the neural network. New items and characters have been successfully introduced. Both pre-existing characters and imported characters are able to interact with all objects with greater effectiveness, responsiveness and visual fidelity.

**Keywords:** Computer graphics · Animations · Recurrent neural network

## 1 Introduction

In recent years, the interest in virtual character animation has rapidly grown. Application areas that benefit from advances in motor synthesis include robotics, virtual reality, video games, and non-interactive animation [6]. The advances in this field are strongly linked to the development of artificial intelligence, especially deep learning [11,15].

Examples of synthesis of human animations aimed at precision interactions can be found in commercial games such as *Gears of War 4* [1], *For Honor* [4] and *Star Citizen* [8] and underline the strong interest of large companies to research in the sector. The main component of the games is immersion, or the


L. T. De Paolis et al. (Eds.): AVR 2021, LNCS 12980, pp. 118–134, 2021.
https://doi.org/10.1007/978-3-030-87595-4_10

user's illusion of not being in front of a machine but of interacting with a real and coherent world. Artificial intelligence plays a fundamental role in this mechanism, not only in the more direct application of creating adversaries and more skilled allies. In particular, the 10 categories identified are: *Non-player character (NPC) behavior learning* [14], *Search and planning* [7], *Player modeling* [22], *Games as AI benchmarks* [20], *Procedural content generation* [18], *Computational narrative* [2], *Believable agents* [13], *AI-assisted game design* [21], *AI in commercial games* [3]. Concerning this last point, in academic field, there is a strong emphasis on the use of Recurrent Neural Network (RNN) for the synthesis of novel animations and transitions starting from Motion Capture (MoCap) data. RNN has good performance to classify sequential data as natural speech recognition, handwriting recognition and animation synthesis. The attempts to increase generality are based on the variety of character actions, the variety of objects they can interact with, and transitions. To do this, it is often proposed to retrain the network with novel data. This process is very expensive for two reasons: first of all, novel data with the same structure as the pre-existing ones are necessary. Tools and methods for acquiring MoCap data are not easily accessible. Data must be cleaned and augmented before being inserted into the training set. Then, the training of the network requires much more time how much more data are available.

This research aims to propose a non invasive method to generalize the techniques of motor synthesis in terms of diversity of build and variety of objects. Departing from the solution proposed by [19], the purpose is to test the introduction of new assets. Through the definition of import rules and the introduction of components in the post-processing phase, the problems of compatibility, visual fidelity and responsiveness have been solved. The components are non-invasive as they do not modify the original structure and are activated only when a correction is needed. Furthermore, the results were obtained without modifying the neural network.

The main contributions of this paper are summarized as follows:

- Visual fidelity: the gap between the hands and the objects carried by the character must be eliminated.
- Effectiveness and responsiveness: characters must lift objects without blocking and without lingering in the lift transition.
- No intervention on the network: the neural network must be unaffected, so corrections cannot be based on introducing novel data or retraining the network.
- Control: activation of fixes must be under the control of the developer.

The remainder of the paper is organized as follows. In Sect. 2, a comparative analysis with other motor synthesis techniques is proposed. Section 3 describes the specific objectives, the problems encountered and the techniques adopted. The results obtained are described in Sect. 4. Finally, Sect. 5 describes the degree of achievement of the objectives and the ideas for future developments.

## 2 Related Works

Several approaches have been adopted to generate realistic animations for humanoid characters. These methods are mainly divided into [10,16,23]:

- *Simulation based*: aims to create animations that respect physical constraints.
- *Data driven*: aims to create animations that can be managed in real time starting from unstructured data.

Since the solution adopted in this paper concerns the data-driven based models with phase-labeled deep-learning, we present literature works for the second approach and the differences with our solution.

### 2.1 Data Driven Models

The purpose of data driven models is the creation of animations and transitions that can be controlled in real time, starting from MoCap data. A classic approach to data driven animation has been proposed by [9] and based on "motion graph". It is possible to control a character by selecting the task, sketch of a path in a maze or by imitating a person in front of a camera. The effectiveness of this solution depends on the quality and the quantity of the available MoCap data. The data are organized in a 2-level structure (motion graph). The top level is a probabilistic mode which groups character states into clusters and serves as an interface for user control. The lowest level is a Markov model which handles the transition between animations. At the lowest level, data are organized in trees, with frame precision. By traversing the trees, the smoothest transition to the next frame is determined. This solution works when the character slowly moves and for a database of comparable size to that used in the experiment.

The work of [17] proposes a motor for the synthesis of constrained movements, offline (not interactive) based on motion graph. The user can specify a high level goal. The engine is capable of generating a realistic offline animation. The traversing of the motion graph is performed through an algorithm that allows to specify an inflation factor for the heuristic that decreases the traversal times of the tree, to find sub-optimal solutions. By compressing and pruning the shaft, the difference between suboptimal and optimal solutions decreases. Animations of 15 s were successfully created by specifying 2d trajectories and applying various constraints.

In time, data driven models have shown encouraging results by adopting machine learning techniques, in particular reinforcement learning.

In [12], the authors developed a controller based on reinforcement learning, powered by a first-person environmental sensor and a motion graph containing MoCap data. The sensor organizes the data in a hierarchical structure and does not require parameterization. The character is able to perceive the depth of the scene and the obstacles that surround it. Planning is formulated as a Markov decision problem. For training, a reinforcement learning algorithm is used, adapted to benefit from the hierarchical structure of the data. At the end

of the training, the controller is able to manage the animations of the character in real time and avoid obstacles.

Deep learning techniques have shown an excellent scalability and ability to create credible unpublished animations. In particular, the RNNs have proved to be very efficient in managing animations. The RNNs in fact take into account the previous states in the prediction phase, for this reason they present good performances for sequential tasks such as natural language processing, speech recognition, handwriting recognition or animation synthesis [5,19]. In [5], the authors have proposed a prediction and classification technique of humanoid poses based on Encoder-Recurrent-Decoder, a particular implementation of RNN. This model operates on two domains: MoCap data and video footage.

The reference solution adopted [19] belongs to data driven models based on deep learning. This model, based on a novel architecture called NSM, allows to start from MoCap data to obtain a character that can be controlled in real time. Novel transitions not present on the original data are generated, precise interactions can be made, and the character can be controlled through high-level tasks. The functioning of NSM consists of two main components. i) Gating Network generates weights (experts) used to modify the value of the internal weights of the Motion Prediction Network. It receives a subset of the Motion Prediction Network inputs. ii) Motion Prediction Network is the main component that deals with the prediction of the position to the next frame.

Similar to the [12] technique, the character is equipped with sensors. The interaction sensor activates when the character has to sit or lift an object and detects the geometry of the target. It is shaped like a parallelepiped and is centered on the object. The environmental sensor is centered on the character, has a cylindrical shape, is always active and informs the network about all the near obstacles. The advantages of this solution compared to the previous ones are: interactions of precision, generalization and response and control times.

The solutions shown are often specific to a certain character model. In some cases, no mesh is assigned to the characters. Attempts to extend generality are oriented towards the environment, the tasks that can be performed and the variety of objects to interact with. The expansions are carried out upstream, affecting the variety and quality of the training data, and thus retraining the network. No attempts have been made to expand the solutions to characters of different sizes.

This research has showed an increase in generality compared to the solution of [19] without intervening on the training of the network. The solution adopted is based on the addition of non-invasive components in the post-processing phase. This approach has 2 advantages: non-invasiveness and no training.

## 3 Materials and Methods

This section shows the solution approaches adopted for the 2 main procedures:

- Adaptation of the Carry task;
- Adaptation of the Sit task and positional corrections.

For each approach we describe the problems occurred in the original design and the adopted solution. The common purpose is to augment the generality of the solution, allowing the use of novel objects and characters and to increase the visual fidelity of the interactions.

*Import of Custom Assets.* During the analysis of the demo, the need derived to import a custom character, as the included character had display problems. The import of custom characters, such as the import of objects to interact with, is a relevant aspect if the intent is to evaluate the versatility of the animation tool in adapting it to different projects. Moreover, the solutions adopted in this study provide good performance also when the characters have been imported.

### 3.1 Adaptation of the Carry Task

One of the actions that the character can perform once it is set up is to carry objects. For that regard, this section highlights the defects, describing and showing the problems related to the transport and their respective causes. In addition, in order to understand the corrections, the steps of the algorithm of lifting at threshold, which manages lifting and transport, was analyzed. Changes of thresholds are also shown to improve effectiveness and responsiveness. Finally, two approaches were used to eliminate the empty space between the hands and the object, the first modifies the rotations of the character's bones while the second modifies the geometry of the object.

The imported character has some problems:

- Effectiveness: some failures in lifting objects. In some cases, when the character is preparing to lift the object, it remained in the position for a very long time before starting the lifting or in the worst cases it blocked without success.
- Efficiency: unacceptable response times. Since the project is intended for interactive applications, such a lack of responsiveness is unacceptable.
- Visual fidelity: the hands are too far from the raised object. The cause of the distance of the hands is given by the difference between the skeletons of the default character and the imported character. The network performs better with the default character, as because it also has longer arms and a mesh that hides the defects when present.

**Algorithm of Lifting at Threshold.** The algorithm of lifting at threshold is structured in 5 main phases.

1. Approaching the object: if the character is not close to the object, he proceeds by walking to a target position.
2. Lifting: the character is in the target position and is ready to lift the object. Bending down, he grabs the object with his hands, then stands up to carry it.

3. Movement with object in hands: it is possible to control the character to make him walk while carrying the object.
4. Positioning transition: when the transport signal is interrupted, the transition is started to place the object on the ground.
5. Positioning of the object on the ground: the object is placed on the ground and the Carry task ends.

In the approaching phase, at each iteration the character continues walking towards the object. The interaction sensor is constantly updated.

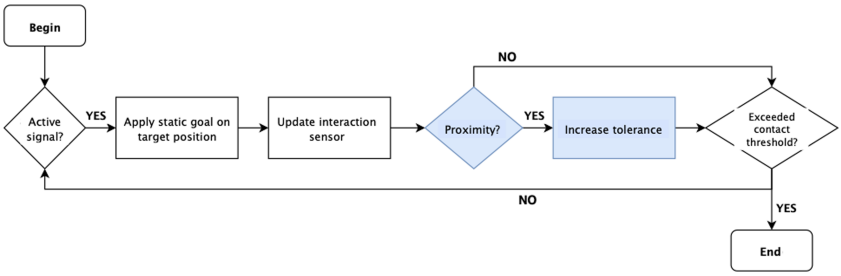


**Fig. 1.** The modified approaching phase (changes are highlighted in blue). (Color figure online)

To solve the problem in the approaching the object, the proposed approach acts on the contact threshold (Fig. 1). The lock of the character occurred at the check on the contact threshold. Even if the character was close enough to the object with the hands in the correct position, the neural network did not assign contact values such as to allow him to exit the cycle. The introduced tolerance increase mechanism is triggered only if the character is already bent to lift the object and the tolerance increases with each iteration both towards the hand-object distance and the contact values.

Even if there are no changes in the lifting phase, the updates of the threshold values affect the stability of the cycle, when checking the contact values. At each iteration, the time elapsed from the beginning of the lift, the position of the object, the static goal and the interaction sensor are updated. Once the lift limit time is exceeded, the next phase begins. It is possible that this phase prematurely ends due to a lack of contact or interruption of the signal. In this case the character drops the object to the ground.

The modify in the movement phase (Fig. 2) provides that before entering the main loop it is signaled that the character is standing and carrying the object. This allows to isolate when the change in the position of the hands is necessary.

For each iteration, the destination of the object is calculated, i.e. the middle position between the hands, which is then assigned to it. Following the application of the dynamic goal, the character can be made to move through movement signals. Finally, the interaction sensor is updated. This phase ends only when the

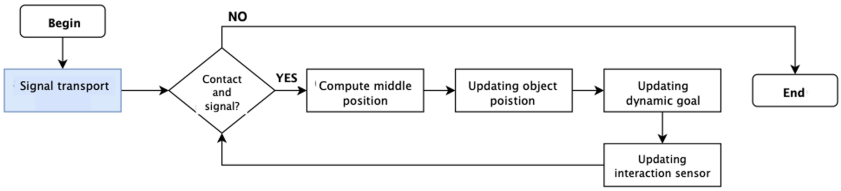


**Fig. 2.** The modified movement phase.

transport signal is interrupted and its duration therefore depends on the user's control.

For the transition phase (Fig. 3), the modify concerns the reset of the contact threshold, so that the character does not stay in the transition phase for too long. Before the change, the threshold values were not modified so the reset was not necessary.

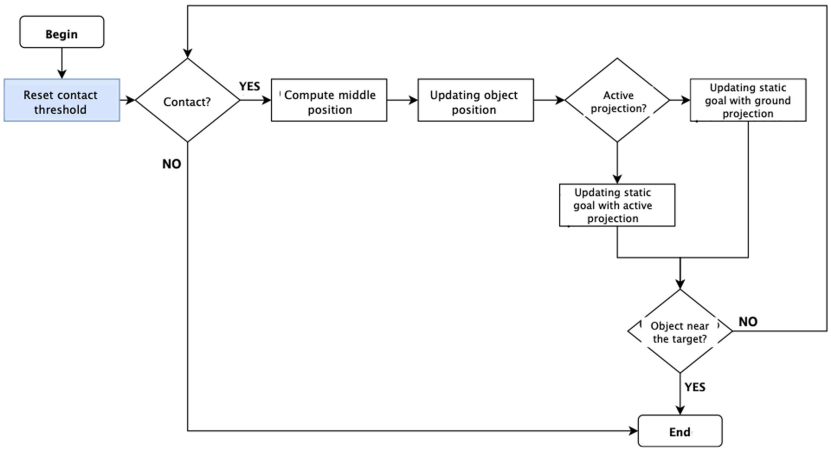


**Fig. 3.** The modified transition phase.

For the final phase of the repositioning of the object on the ground (Fig. 4), the modify concerns the control of the distance of the object from the final position. If it is close enough then the transition phase ends.

**Wrist Correction.** In some cases the character's hands are too far from the transported object. This defect was both in the imported character and the one included in the demo. To contextualize the change it is necessary to understand the logic of assigning the character's posture (Fig. 5). The network is powered for each frame, in order to be able to predict the values.

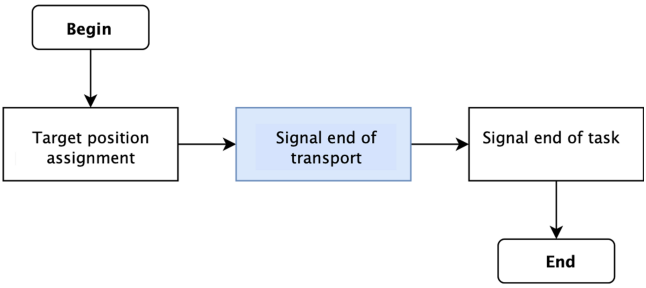


**Fig. 4.** The modified repositioning phase.

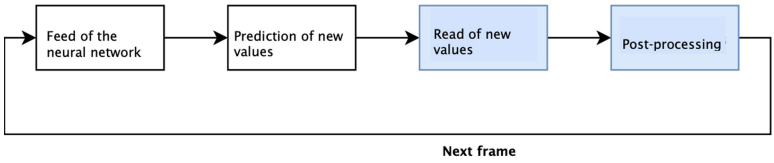


**Fig. 5.** The modified update phase for each frame.

During the read of new values, the positions and rotations of the bones for the current frame are assigned. In the Postprocess phase corrections are made to the skeleton through assignments and calculation of inverse kinematics. Figure 6 shows the Read phase. After updating the time series and the trajectory, the posture is assigned. A check has been added that allows to disable the network control on the skeleton, in order to facilitate tests and isolate conflicts.

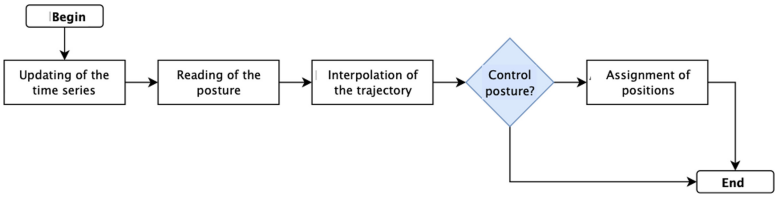


**Fig. 6.** The modified reading phase.

Three strategies have been adopted to correct the position of the wrists.

1. Intervention on transformations: in the postprocessing phase the transformations of the wrists were modified. The first attempt was to intervene directly on the positions of the wrists, then on the rotation of the elbows.
2. Intervention on colliders in the editor: by intervening on the objects transported incorrectly and modifying the geometry of the colliders, the network is influenced to bring the hands closer to the object.

3. Intervention on colliders at runtime: the modification of the colliders is carried out in real time by comparing the position of the hands with the extension of the object.

**Adaptation of the Sit Task.** The Sit task consists of making the character sit on the top of an object. The complete interaction includes 4 phases: i) Approaching the object; ii) Transition into a sitting position; iii) Permanence in a sitting position; iv) Transition into a straight position (Idle).

The problem concerned a malfunctioning in staying in a sitting position. In particular, the ankles were not positioned correctly and interpenetrations between the mesh of the character and the object were visible. The different length of the legs creates an inconsistency at the moment of arbitrary positioning of the ankles in the Read and Postprocessing phase. Since the network expects longer legs, it places the hips too deep, causing the mesh to interpenetrate. To solve the problem, there has been a change in the algorithm. By intervening in the algorithm it is sufficient that the modification is carried out after the inverse kinematics so that the old position is overwritten (the final position is the one that counts for rendering).

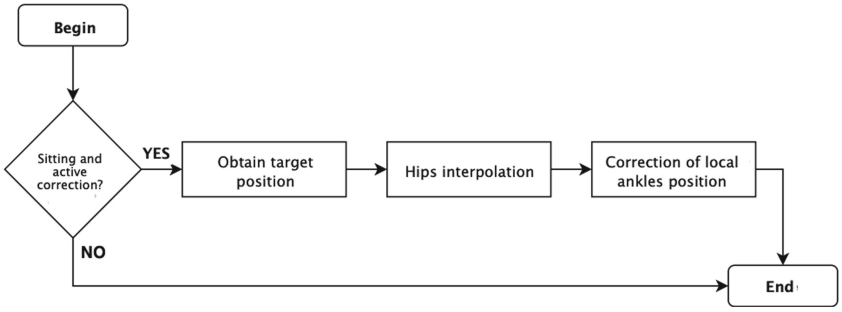


**Fig. 7.** Correction in Post-process of the task Sit.

Figure 7 shows the correction of the Sit task. The correction is activated only if the character is seated and if it is activated for the object with which he is interacting. All objects that the character can sit on have a contact point for the hips. To allow a smooth correction, an interpolation is performed with the desired position. Finally, we intervene on the local position of the ankles to eliminate graphic artifacts.

**Positional Corrections.** The default potion is also a task. If the character did not receive any assignment to the skeleton positions while standing still, she would initially remain in T-pose. The animation in the Idle condition gives the character a natural look.

The problem is that even without receiving any input, the character tends to move around the scene. The skeleton remains motionless while the entire character moves along the surface creating a sliding effect. The movement of the character is given by the assignment to each frame of the new position of the bones. The position at each frame is determined by the neural network. For each bone there are two network outputs that directly affect its new position, the positions of the bones and their speeds (Fig. 8).

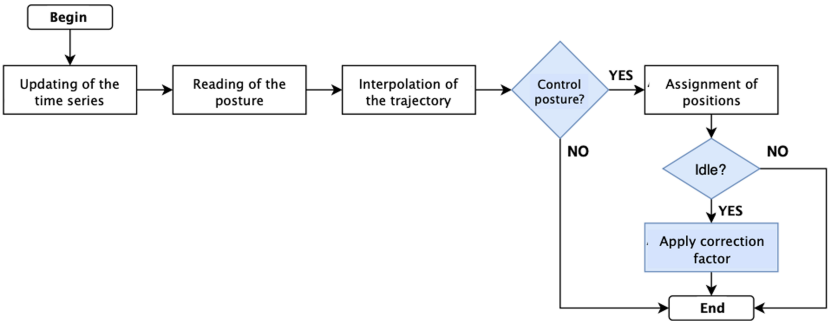


**Fig. 8.** Modified post-process phase to correct sliding.

When the posture reading is taken, position and speed are calculated for each bone.

The calculation of the actual position is $pos_f = \frac{(pos_{f-1}+velocity \cdot time)+pos}{2}$, where $pos_f$ is the position in the current frame, $pos_{f-1}$ the position in the previous frame, *velocity* the speed obtained from the network and *time* the elapsed time. Sliding is corrected by applying a "correction factor" to the speed of each bone, only when the character should be standing still.

## 4 Results

This Section shows the results obtained for the 2 areas of improvement: adaptation of the Carry task and adaptation of the Sit task and positional corrections.

### 4.1 Adaptation of the Carry Task

The success criterion is given by the correct lifting of all objects (pre-existing and new) both by the original character and the imported ones. By applying the appropriate correction method it was possible to make all the characters lift all the objects without creating gaps.

The 3 methods adopted were:

- Intervention on the transformations;
- Intervention on the colliders in the editor (sensor);
- Intervention on the colliders at runtime (sensor).

**Intervention on the Transformations.** The first intervention on the transformations was on the position of the wrists. The effect of this correction can be seen in Fig. 9. Although the hands are in a believable position relative to the carried object, the mesh associated with the elbow has an incorrect rotation. By forcing arbitrary positions in the skeleton, it is possible to create errors of this type, as inconsistencies in terms of lengths and rotations can occur. The network's assignment of global positions takes the wrists away from the correct local position.

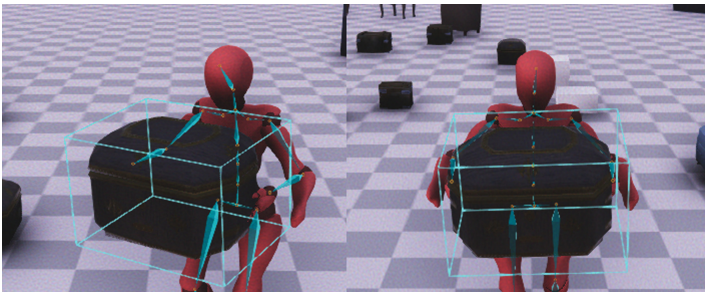

**Fig. 9.** Effect of direct positioning.

The intervention on the rotation is the most solid, as it help to position the hands correctly and partially returns control of the skeleton to the developer. This aspect allows more freedom in the positioning of the bones, but can generate conflicts with the network. These conflicts can show as graphic imperfections in some characters. For the conflicts with the positioning of the neural network, we have decided to intervene directly on the values detected to influence the rotation of the arms, and therefore leave the network for posture complete control again. Visual problems can also be related to the character skinning process. To solve these problems it is therefore necessary to go back to the "weight painting" phase, uniform the weights of the mesh and then re-import the character.

**Intervention on the Colliders in the Editor.** Figures 10 and 11 shows the effects of the resizing of bounds and colliders along the X-axis. The objects are always raised considering the forward direction, which coincides with the Z-axis (in blue). The space between the hands and the object is eliminated after the modification.

To determine if and when to apply the scaling, the same control considered for the rotation of the elbows was used. It is then checked that the character is actually being transported and that the object contains the contact points of the wrists. The intervention on the colliders in the editor allows to have a good control for a specific object, leaving unchanged the interactions with the other objects. Since the positions and rotations of the bones are not arbitrarily changed, no conflicts with the network are generated. This method requires more

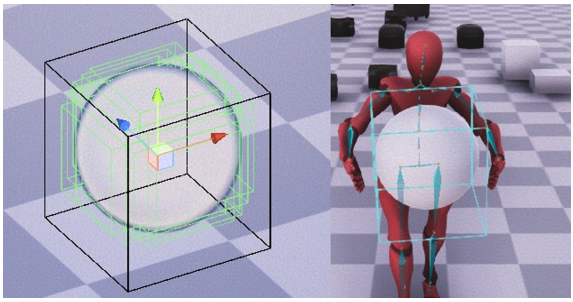

**Fig. 10.** Collider, bounds and effect on transport before the modification. (Color figure online)

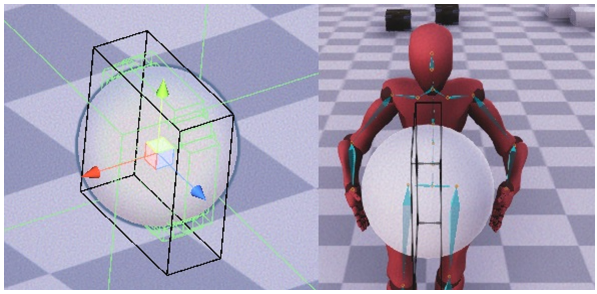

**Fig. 11.** Collider, bounds and effect on transport after the modification. (Color figure online)

effort on the part of the developer who has to test the transport of the object and make any changes to the colliders. Another disadvantage is the specificity, as the modification to the colliders is static, it may be correct for a certain character but not for others. With multiple characters this approach may not work. Another limitation of this approach compared to direct intervention is that it fails to correct the transport of smaller objects (Fig. 12).

**Intervention on the Colliders at Runtime.** Figure 13 shows the result of this operation equal to that obtained through the modification in the editor. The intervention on colliders at run-time, exploiting the difference between the distance of the hands and the extension of the object, can automatically adapt to different characters and geometries. It therefore has a good generality and automaticity. The intervention on the sensor does not generate conflicts with the neural network, but could have conflict with corrections made offline. For this reason, an option has been added in the Interaction component that allows to disable corrections for certain objects.

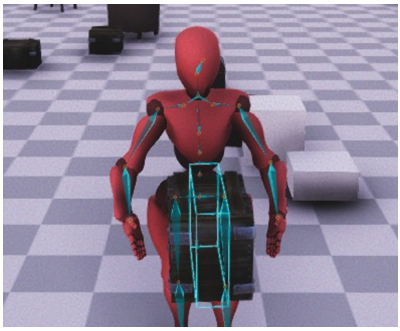

**Fig. 12.** Example of inefficiency with too small objects.

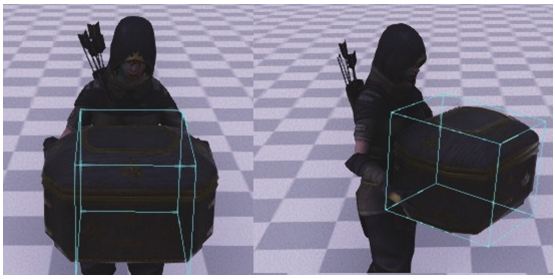

**Fig. 13.** Effect of the fix at runtime on a new character. The size of the bounds (in blue) is restricted with respect to the mesh.

**Comparison.** The comparison among the 3 described methods can be summarized in the Tables 1 and 2.

Thanks to a selector, it is possible to easily change the correction method.

### 4.2 Adaptation of the Sit Task and Positional Corrections

In the Sit task, the visual fidelity is increased thanks to the correct positioning of the ankles and the elimination of interpenetrations. The autonomous sliding has been minimized and the character has been brought back to the ground via the height correction.

**Table 1.** Summary table of the 3 correction methods (1).

| Method | Generality (Characters) | Generality (Objects) | Control | Conflict with the neural network |
|---|---|---|---|---|
| Rotation of the elbows | High | High | Medium | Yes |
| Offline modification | Low | Low | High | No |
| Run-time modification | High | Low | Medium | No |

**Table 2.** Summary table of the 3 correction methods (2).

| Method | Automaticity | Integrity | Problems |
|---|---|---|---|
| Rotation of the elbows | Yes | High | Problems with the mesh of the character |
| Offline modification | No | Media | Does not manage good little objects |
| Run-time modification | Yes | Low | Does not manage good little objects |

**Sit Task.** The results are shown in Fig. 14. The detachment with the ankles was canceled. In the chair, the contact points have been changed to allow the character to sit further forward. The knees are positioned at the edge of the chair avoiding interpenetration. However, this translation is in a less natural arm position. Another defect is the loss of fluidity in the transition: in the most serious cases it is possible to notice a sort of teleportation of the character to the target position. Finally, as in the case of the transport, the corrections are specific to the character and mainly depend on the length of the legs. In many cases, the fix to the Sit task is unnecessary and the control can be left to the network. Multiple touch-points can also be defined for different characters.

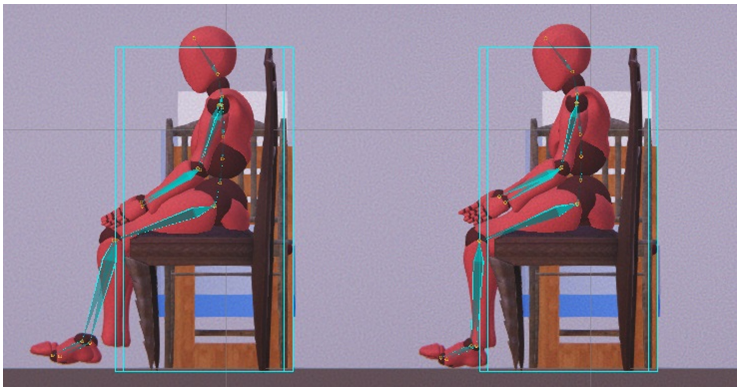

**Fig. 14.** Comparison before (left) and after (right) the correction.

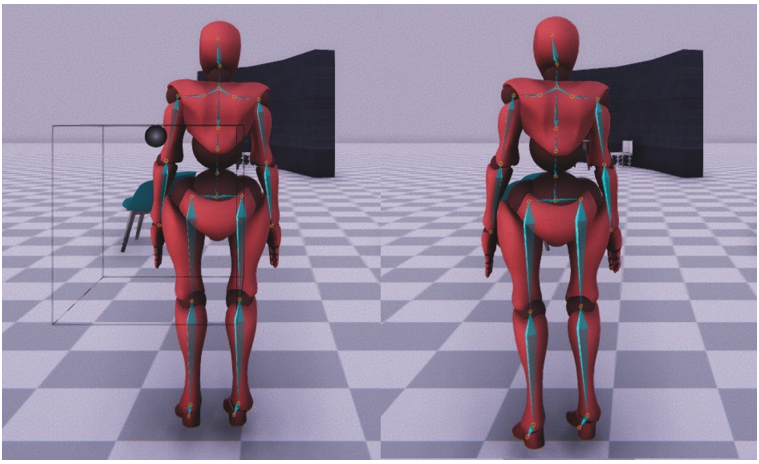

**Fig. 15.** Timelapse of 30 s without input with corrective factor.

**Table 3.** Comparison of translation in the absence of input and in the absence or presence of the correction factor in 30 s.

| | Translation X | Translation Z | Absolute translation |
|---|---|---|---|
| Without correction | –0.5 | +1.4 | 1.49 |
| Active correction | –0.2 | –0.1 | 0.22 |

**Sliding.** The sliding correction by acting on the speeds greatly improves the stability of the character (Fig. 15). A more solid (and more invasive) solution would be an integral modification of the system by which the translation in the scene is established. The effect of the correction is shown in Table 3.

## 5 Conclusions and Future Works

The use of RNNs to generate animations despite the high success in the scientific field and the excellent results in real development still has limitations that restrict its use on a larger scale. In particular, the lack of generalization of the networks and therefore the difficulty of their use on computer graphics characters of different nature or composition, studies are needed to overcome this limit. Furthermore, the composition of animations and interactions with the scene, based on the data, presents graphical-visual renders in some cases not exactly realistic which prevent their application in areas where the visual rendering of the result is of fundamental importance, such as in games. To overcome these limitations, this work has focused on the phase following the generation of interactions by the network. In particular, starting from the work of [19], its limitations were analyzed and solutions were proposed that made it possible to improve the visual rendering of the Carry and Sit operations.

The proposed correction method is non-invasive. Corrective factors have been added, both in the logic of the task and in Postprocess, which improve visual fidelity, effectiveness and response times. In addition, new characters and new objects have been successfully introduced to make them interact. The new objects are compatible with the existing solution. The improvements made are also reflected in the starting assets. The original character is more responsive and can correctly lift all objects. Additional components have a good tradeoff between automaticity and control. Thanks to the introduction of control variables, they can be easily activated, calibrated and tested during execution. The results have been obtained without modifying the neural network. This is important as it demonstrates that it is possible to make minor changes to some network inputs without creating conflicts. It is also possible to avoid the operations of new data acquisition, preparation and training. The objectives have been fully reached and an increase in the generality of the solution has been achieved.

Possible future developments may concern two aspects:

- Collision management: the character can avoid obstacles thanks to the environmental sensor. He can pass through cracks by bending over or sit on a chair behind a desk by moving his legs appropriately.
- Importing IK characters: by resolving IK constraints at runtime, offset errors between the bones can be corrected. To obtain this advantage, the imported character must have been configured with inverse kinematics. It is therefore interesting to study the behaviour of IK characters in lifting objects and sitting down. The application of inverse kinematics has been included in the correction by rotation of the elbows and in the Sit task.

## References

1. Bollo, D.: High performance animation in gears of war 4. In: ACM SIGGRAPH 2017 Talks. SIGGRAPH '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3084363.3085069
2. Burelli, P., Yannakakis, G.: Combining local and global optimisation for virtual camera control, pp. 403–410 (2010). https://doi.org/10.1109/ITW.2010.5593328
3. Buro, M.: Real-time strategy games: a new AI research challenge. IJCAI **2003**, 1534–1535 (2003)
4. Clavet, S.: Motion matching and the road to next-gen animation. GDC Vault, New York, NY, USA (2016). https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road
5. Fragkiadaki, K., Levine, S., Felsen, P., Malik, J.: Recurrent network models for human dynamics. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 4346–4354 (2015)
6. Frontoni, E., Loncarski, J., Pierdicca, R., Bernardini, M., Sasso, M.: Cyber physical systems for industry 4.0: towards real time virtual reality in smart manufacturing. In: De Paolis, L.T., Bourdot, P. (eds.) AVR 2018. LNCS, vol. 10851, pp. 422–434. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95282-6_31
7. Hazim, N., Al-Dabbagh, S.S.M., Naser, M.A.S.: Pathfinding in strategy games and maze solving using a* search algorithm. J. Comput. Commun. **04**, 15–25 (2016). https://doi.org/10.4236/jcc.2016.411002

8. Herzog, I.: Teaching a character how to walk with procedural assisted animations. CitizenCon (2017). http://youtube.com/watch?v=ZMgHhu7XT78
9. Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K., Pollard, N.S.: Interactive control of avatars animated with human motion data. ACM Trans. Graph. **21**(3), 491–500 (2002)
10. Li, Z., Zhou, Y., Xiao, S., He, C., Huang, Z., Li, H.: Auto-conditioned recurrent networks for extended complex human motion synthesis (2017)
11. Liciotti, D., Paolanti, M., Frontoni, E., Zingaretti, P.: People detection and tracking from an RGB-D camera in top-view configuration: review of challenges and applications. In: Battiato, S., Farinella, G.M., Leo, M., Gallo, G. (eds.) ICIAP 2017. LNCS, vol. 10590, pp. 207–218. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70742-6_20
12. Lo, W.Y., Knaus, C., Zwicker, M.: Learning motion controllers with adaptive depth perception. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '12, Eurographics Association, Goslar, DEU, pp. 145–154 (2012)
13. Loyall, A.B.: Believable agents: Building interactive personalities. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, Technical report (1997)
14. Lucas, S.M., Kendall, G.: Evolutionary computation and games. Comp. Intell. Mag. **1**(1), 10–18 (2006)
15. Paolanti, M., Frontoni, E.: Multidisciplinary pattern recognition applications: a review. Comput. Sci. Rev. **37**, 100276 (2020)
16. Park, S., Ryu, H., Lee, S., Lee, S., Lee, J.: Learning predict-and-simulate policies from unorganized human motion data. ACM Trans. Graph. 38(6) (2019). https://doi.org/10.1145/3355089.3356501
17. Safonova, A., Hodgins, J.K.: Construction and optimal search of interpolated motion graphs. ACM Trans. Graph. **26**(3), 106-es (2007). https://doi.org/10.1145/1276377.1276510
18. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-42716-4
19. Starke, S., Zhang, H., Komura, T., Saito, J.: Neural state machine for character-scene interactions. ACM Trans. Graph. **38**(6) (2019). https://doi.org/10.1145/3355089.3356505
20. Xia, B., Ye, X., Abuassba, A.O.: Recent research on ai in games. In: 2020 International Wireless Communications and Mobile Computing (IWCMC), pp. 505–510. IEEE (2020)
21. Yannakakis, G.N., Togelius, J.: A panorama of artificial and computational intelligence in games. IEEE Trans. Comput. Intell. AI Games **7**(4), 317–335 (2015)
22. Yannakakis, G.N., Spronck, P., Loiacono, D., André, E.: Player modeling (2013)
23. Zhang, H., Starke, S., Komura, T., Saito, J.: Mode-adaptive neural networks for quadruped motion control. ACM Trans. Graph. **37**(4) (2018). https://doi.org/10.1145/3197517.3201366