Contrôle du mouvement de marche de personnages virtuels en milieu liquide

Samuel Carensac

Département informatique
INSA de LYON
2014/2015

Sous la responsabilité de :
nom du tuteur : Saida Bouakaz, Nicolas Pronost
nom de l'enseignant : Stéphane Bres

Résumé L'animation basée physique est de plus en plus étudiée car elle permet de réaliser les interactions avec l'environnement beaucoup plus naturellement. Bien que certains contrôleurs de mouvement permettent la simulation des interactions d'un personnage en milieu liquide ceux-ci ne simulent que la nage. Nous présentons une stratégie de contrôle de mouvement de marche d'un personnage virtuel en immersion partielle dans un liquide. L'effet des liquides sur le mouvement du personnage est modélisé à l'aide de formules d'hydrodynamiques simples. Notre contrôleur permet de combiner différents styles de marches, d'assurer l'équilibre par le placement du pied et d'assurer le contrôle précis de la vitesse du personnage. Nous avons déterminé les paramètres optimaux pour divers critères d'évaluations à l'aide d'une phase d'optimisation. Cette optimisation a été effectuée sur cinq niveaux de liquide régulièrement répartis. À la suite de cette optimisation nous avons conçu un contrôleur capable de s'adapter automatiquement aux variations de niveau d'immersion, de densité du liquide et de vitesses. Notre contrôleur est également robuste à l'application de forces externes.

Keywords: Humain virtuel, Animation basée physique, Contrôle de mouvement, Optimisation hors ligne

Abstract. Physics-based animation is an increasingly studied subject of computer animation because it allows natural interactions with the virtual environment. Though some existing motion controllers can handle the simulation of interactions between a character and a liquid, only few methods focus on the simulation of the locomotion of immersed bipeds. In this paper, we present a control strategy capable of simulating partially immersed gaits. The impact of the liquid on the character's motion is modeled through simple hydrodynamics. To produce natural looking animations, we design a controller allowing the combination of multiple gait styles, the conservation of balance through foot placement and precise control of the character's speed. We determine the optimal parameters for the controller by using an optimization process. This optimization is repeated for several scenarios where the character has to walk across a volume of liquid parametrized by its height. Our controller produces natural looking gaits while being capable of online adaptation to the variation of liquid height, to the modification of the liquid density and to the variation of the required character's speed.

Keywords: Virtual Human, Physics based animation, Motion control, Offline optimization

1 Introduction

Simulating realistic human motion is a key step in creating virtual environments. Over the years, these environments have become increasingly diverse and complex with a large number of elements

that can influence the motion of a virtual character. In those cases, physics-based animation is preferred over kinematic animation since it does not need a series of exact position for each possible interaction. With this increasing complexity, it is difficult to achieve realistic interactions between animated characters and the environment using kinematics-based approaches. Physics-based animation uses physical phenomena (forces and torques) to manipulate the character. This allows the creation of a motion that will be directly impacted by the environment. A growing number of contributions are now working on building controllers using simulated physics [Geijtenbeek and Pronost, 2012]. Although they inherently allow obtaining interactions with the environment, the manipulation of the character becomes more complex as no direct control over the position of the limbs of the animated characters is possible.

The main challenge of a controller is to allow high level parametrization of the system. For example, these high level parameters may be the character's speed, the direction of displacement or the motion style. The need to simulate a large number of motion styles (walking, running, jumping...) makes creating a generic controller extremely challenging. Similarly, the mastery of multiple interactions with the environment also increases considerably the complexity of the system. This is why the existing controllers focus on the study of a limited number of motion styles at a time and interactions with the environment [Geijtenbeek and Pronost, 2012]. We focused on the control of walking in partial immersion in a liquid. Our objective was to define and implement the necessary mechanisms to a physics-based controller to allow real-time animation of a virtual character interacting with a liquid. Our controller is capable of great freedom of gait style and precise tracking of the character's motion speed.

The remainder of this paper is structured as follows. Section 2 reviews previous works. Then in section 3, we give an overview of our system. Sections 4 to 7 describe the tools used in the implemented system. Section 8 illustrates a variety of results, provides discussions and gives the limits of our method. In section 9, we summarize our approach and highlight future areas of work.

2 Previous works

Numerous works on control of virtual characters with simulated physics can be found in the literature [Geijtenbeek and Pronost, 2012]. Among those works, some share common characteristics with our objectives.

Our work share some features of the SIMBICON (SIMple BIped CONtroler) [Yin et al., 2007] and associated works. Among them, [Coros et al., 2009] propose a system integrating multiple controllers for navigation tasks. The drawback is that the system is designed to use optimization to determine how the different controllers should be used. To enable the use of low gains in the PD-controllers, different methods have been used such as a feedforward system [Yin et al., 2007] or the computation of torques to compensate the effect of gravity [Coros et al., 2010].

Balance control is one of the key systems in physics-based simulation. For instance in [Yin et al., 2007], secondary PD-controllers are placed on the key joints (stance ankle and swing hip) to dynamically adapt the tracked positions. One way to compute a balance aware foot placement during walk motion is to use an Inverted Pendulum Model (IPM) [Coros et al., 2010; Kajita et al., 2001]. This model can be used to dynamically compute the trajectory of the swing foot but it limits the range of possible gait styles.

Velocity control is often one of the required characteristics of physics-based controllers. Some systems are capable of adapting to online variations of the target velocity. In [Coros et al., 2009], multiple controllers are combined for specific motions as forward and backward walks. The IPM based systems offer an inherent control though imprecise as it does not consider the observed velocity [Coros et al., 2010]. The use of horizontal virtual forces has been a way to obtain a precise control of the speed and balance of the character [Coros et al., 2010; Geijtenbeek et al., 2012]. This system is based on secondary PD-controllers to compute the requisite force. The static gains and constants used in those systems make them unable to track correctly the target velocity when the character is subject to a variable environment (e.g. apparition of a liquid medium).

Movement in liquids have been studied in both simulation [Yang et al., 2004; Si et al., 2014] and biomechanics [Barela et al., 2006; Chevutschi et al., 2009]. Those works mainly focus either on swimming control or on gait analysis with high level of immersion, making them unusable in our scenarios. One can also mention [Lentine et al., 2011] who simulated human walk under wind

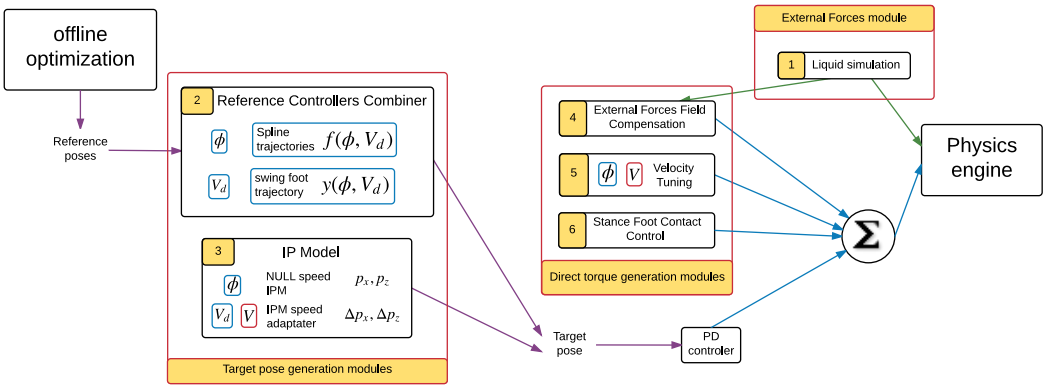

Fig. 1. System overview. (1) influence of liquid through external forces; (2) a combination of controllers generating the target poses; (3) an IPM for predictive foot placement; (4) an external force-field compensation helping the PD-controller; (5) velocity tuning for fine corrections of velocity and balance; (6) an active control of the stance foot contact with the ground. ϕ is called the phase, and is computed by normalizing the current time t since the last footstrike by the expected step time T: $\phi = \frac{t}{T}$.

forces. Most of those works use the Navier-Stockes equations to simulate the liquids making them usually ineffective to obtain real-time and interactive simulations.

To overcome these limitations, we propose a novel controller showing the following characteristics:

- dynamic gait style adaptation by combining multiple reference controllers
- liquid aware gait styles by specific IPM usage and offline optimization
- precise tracking of target velocity by using learning strategies
- real-time interactive simulation by using simple hydrodynamics to simulate the impact of the liquid on the motion

3 System Overview

Our controller consists of six key components that are classified in three categories (figure 1) according to their output, namely: the target pose, the joint torques and the external forces.

First, we simulate the liquid impact on the character by applying external forces computed by simple hydrodynamics (drag, friction and buoyancy) allowing us to obtain real-time interactions between the liquid and the character (section 4). Our method combines various controllers, each one defining a gait style depending on the simulation conditions (e.g. liquid height, target speed), allowing an adaptive evolution of the gait style (section 5.1). The specified trajectories for the joints composing the swing leg are overridden by the results of an IPM if the character is in the falling phase of a step or if the controller detects a loss of balance (section 5.2).

We augment the PD-controller with torques computed through an external force-field compensation. Our system is a extension of Coros et al.'s gravity compensation mechanism [Coros et al., 2010] and computes a significant part of the necessary torques (section 5.3).

Our controller includes a precise tracking of the target velocity through an adaptive offset on the swing foot position computed by the IPM (section 5.2). Our system presents an improvement of Coros et al.'s fine-scale control [Coros et al., 2010] by considering the intra-step speed variation of the character to compute a more efficient virtual force (section 5.4).

We add a feedback controller on the stance ankle that ensures a correct contact between the stance foot and the ground (section 5.5).

Finally, we use offline optimization to generate the reference poses used by the controller combinator. This optimization defines a gait style specific to a given scenario (section 6).

4 External Forces

Beside the ground reaction forces, the external forces generated in our system are the forces related to the influence of the liquid on the character. We consider two forces that are based on hydrodynamics laws. The first one is buoyancy. We use the well-known equation $F_B = -V_i \rho g$ with V_i the immersed volume of the physics representation of the immersed object and ρ the density of the liquid. The second force is the parasitic drag. We restrict the considered physics phenomena to the form drag and the skin friction, modeling the resulting force F_D using the following equation:

$$F_D = \frac{1}{2}\rho v^2 \times (A_n C_d + S_n C_f) \tag{1}$$

where C_d is the drag coefficient, ρ the fluid density, A_n the cross sectional area and v the relative speed to the fluid. Due to the complexity of dynamically computing C_d, we set it to 1.0 (average value for a man). S_n is the surface in contact with the water and C_f is a coefficient representing the fluid viscosity allowing us a rough representation of the friction. The velocity varying through the limbs, we use a finite elements decomposition of the surface of each limb and compute the parasitic drag for each element individually. The computation of the cross sectional area A_n and the immersion test are both directly realized on each element. We use the same finite elements to obtain an approximation of the surface in contact S_n.

5 Control Framework

Our system is built on the version of SIMBICON presented in [Coros et al., 2010]. The trajectories for the ankles, pelvis, back and head are specified in the coordinate system of the character. This coordinate system is orthonormal, where the z-axis is the sagittal axis of the character, the x-axis is the coronal axis of the character and the y-axis points to the ground. The trajectories for the shoulders, elbows, toes and the knee of the stance leg are specified using local joint coordinates. Unlike [Coros et al., 2010], we allow the user to specify the swing foot trajectory which will give us access to a bigger range of possible motions.

5.1 Reference Controllers Combiner

The goal of the *Controllers Combiner* is to observe dynamic variations of the gait style depending on the conditions of the simulation (e.g. liquid height). Following [Coros et al., 2009], our *Controllers Combiner* is built on multiple reference controllers that are specific to one set of conditions. Even though we limit the conditions to the walking speed and the liquid height, our system could handle other specifications. Each reference controller defines the trajectories for the joints that exhibit significant variations from one standard controller. For example, our standard controller is a forward walking controller. It specifies that the heel hits the ground before the toes do. To walk backward, the user has to specify a reference controller stating that the toes hit the ground before the heel and affect a negative sagittal speed to it. Our reference controllers differ from the ones used by [Coros et al., 2009] by two characteristics. First, we do not require each individual controller to produce a stable motion. In our system, the balance is acquired by the use of an IPM. Second, each individual controller only specifies the joints where the variation from the standard controller is significant. Additionally, our system does not require any optimization step to find the optimal combination of the reference controllers. Instead, when the character ends a step, the system will compute a new trajectory for each joint. Those trajectories are obtained by a square-law interpolation between the two nearest reference controllers. For example, given n reference controllers, we will use the two reference trajectories f_i and f_{i+1} defined for the speeds V_i and V_{i+1} to compute the target trajectories f:

$$f = f_i * (1 - R_i) + f_{i+1} * R_i \quad \text{with} \quad 0 \leq i \leq n$$

$$\text{and} \quad \begin{cases} R_i = \left(\frac{V_d - V_i}{V_{i+1} - V_i}\right)^2 & \text{if} \quad \exists i : V_i < V_d < V_{i+1} \\ R_0 = 0 \\ R_n = 1 \end{cases}$$

The same formula is applied to adapt the gait style to the liquid height where the speeds are replaced by heights.

5.2 Inversed Pendulum Model

We use an *Inversed Pendulum Model* (IPM) supposing constant leg length and zero target velocity similar to the one used in [Coros et al., 2010]. We have modified this model to allow the previously impossible specification of gait styles and to enable a better tracking of the user's target velocity.

Specific IPM Usage Our idea to enable the specification of new gait style is that the IPM does not need to control the swing foot during the whole step. We only need to control the position of the swing foot when the character is in a falling state, meaning when the vertical speed of the center of mass is positive $V_{COM} > 0$. During a step, the falling phase corresponds to the end of the step. So during the first part of the step we will use a user defined trajectory for the swing foot. This allows the observation of vertical movement of the swing foot without any horizontal movement, which was previously impossible. This kind of gait style is important in our scenarios as it is typical of a character trying to minimize the drag from moving in a liquid.

IPM Results Alteration To have a partial control of the velocity, [Coros et al., 2010] use a linear modification of IPM results depending on the target character velocity ($\Delta(x,z) = -\alpha V_d$). Unfortunately it only works properly near the velocity for which the linear factor has been optimized. In particular, this system cannot handle the transition from an environment with a fluid medium to an unconstrained one.

Our solution is to add a supplementary offset $\Delta(x,z)_i$ to the results of the IPM. This offset will be modified at the end of each step s depending on the difference between the current character velocity and the target velocity $\Delta(x,z)_s = \Delta(x,z)_{s-1} + \beta(V - V_d)$ with β a positive constant. The swing foot position from the IP model $P_{IPM}(x,z)$ will therefore be modified as follows:

$$P(x,z) = P_{IPM}(x,z) - \alpha V_d + \Delta(x,z)_s$$

To prevent parasiting between this system and the velocity tuning (section 5.4), we modify the offset only if the virtual force applied by the velocity tuning reaches 80% of the maximum force allowed.

5.3 External Force Fields Compensator

Our *External Force Fields Compensator* is an extension of the gravity compensation proposed in [Coros et al., 2010]. The goal is to dynamically compute a part of the necessary torques at each joint thus permitting the use of lower gains in the main PD-controller. Using virtual forces opposing the external force fields affecting the character gives us a good estimation of these torques. The external forces that we consider are the weight $P = mg$ of each body part of the character and buoyancy $F_B = -\rho V_i$. To prevent time consuming computation resulting from the application of numerous small virtual forces (depending on the precision of the finite elements), we chose to ignore the liquid drag in the compensator. So the final virtual force applied to each body part is:

$$F = -mg + \rho V_i$$

5.4 Velocity Tuning

In [Coros et al., 2010], the controller applies a horizontal virtual force to the center of mass to obtain a fine control of the speed and balance. Our version presents three major differences.

Articuled chain. Instead of considering one chain that goes from the head to the stance foot our system preconises two chains: the first goes from the pelvis to the stance foot and the second only contains the joint between the pelvis and the torso. This modification comes from the following two considerations. First, we do not consider the head in the top chain because we do not want the character to tilt the head to control its velocity. Secondly, we separate the torso from the lower

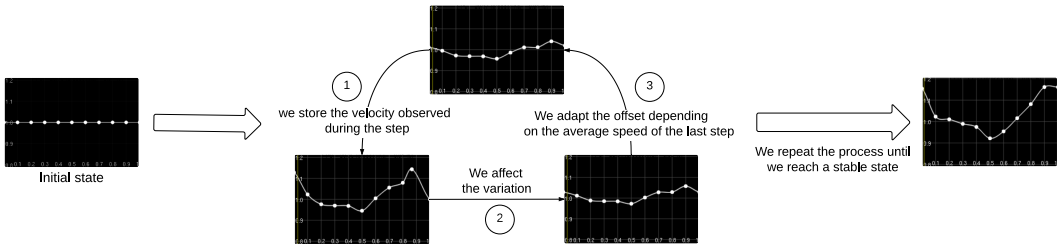

Fig. 2. Process of learning the required velocity curve

body to prevent the character from applying a large torque on the ankle when tilting the torso is enough.

Limbs mass impact. Applying small torques on joints that control the heavier limbs gives a more natural motion than applying important torques to minor joints. Considering this rule we have modified our Jacobian matrix so it takes into account the mass M_i of each individual limb:

$$J_n^T(p) = \frac{1}{M} \sum_{0<i\leq n} ((P_i(x,y,z) - P_{i-1}(x,y,z)) * M_i)$$

where M is the sum of the limbs' mass considered in the velocity tuning, $P_i(x,y,z)$ the position of the i joint and $P_0(x,y,z)$ the application point of the virtual force.

Intra-step velocity variations. In most cases, the velocity is not constant during a step. In most gait styles the character is faster at the start and end of a step than near the middle of it. Supposing it constant as in [Coros et al., 2010] may cause the apparition of undesired behaviors where the controller requests the character to slow down at the start of the step, then speed up near the middle of it, and to slow down once again at heel strike. To limit this issue, we use a system to learn the target velocity at each instant of a step that is needed to produce a virtual force as constant as possible. This constant virtual force would result in the character moving at the global target velocity. The *learning velocity curve* is defined by k key points uniformly distributed. The values between those points are computed using Catmull-Rom splines. Our tests have shown that a value of $k = 11$ gives enough precision without being too time consuming. The learning method is based on an iterative principle (see figure 2). During a step we record the velocity of the center of mass of the character. When the step ends, we check if the average variation Δ_{avg} between the learning velocity curve and the velocity curve observed during the step f_{obs} is lower than a threshold. If so, we move each reference point of the learning velocity curve towards the values read in the observed velocity curve. To help converging without oscillations, we limit the maximum variation that can be affected to each key point to the average variation Δ_{avg}.

To control the velocity of the character we maintain an offset K_{off} for each curve that will be added when reading the values. At the end of each step we add the observed error $Err = V_d - V_{obs}$ to the offset. To accelerate the convergence to a stable state, we added some rules that limit the evolution of the offset depending on how the velocity trajectory has evolved. The most notable rule is to lower the Err by half of the average variation affected to the curve. So, at the end of the step s, each key point $f_s(i)$ is computed with the following equation:

$$\begin{cases} f_s(i) = (f_{s-1}(i) + min(\Delta_{avg}, f_{obs}(i) - f_{s-1}(i))) \\ K_{off}(s) = K_{off}(s-1) + (V_d - V_{obs}) * K_{evo\_speed} \end{cases} \quad (2)$$

With K_{evo_speed} a coefficient set to 1.0 for the sagittal axis and to 0.2 for the coronal axis. If the average variation Δ_{avg} is above the threshold it means the character is not in a stable motion anymore. In that case, we stop adapting our learning curve and switch to the use of recovery steps. This means the controller will then ignore the user specified swing foot trajectory and use the trajectory defined by the IPM for the totality of the step. The controller stays in recovery mode until the average variation between the observed speed and the specified one is lower than the threshold. If it takes more than five steps we reset the learning curve to the constant target value and we start learning again.

5.5 Stance Foot Contact Control

When the character loses its balance or when conditions of the simulation deviate from the ones for the references controllers (coronal velocity, liquids with different characteristics...) the specified trajectories may become inadequate. This problem is most apparent on the stance ankle. An inadequate trajectory will lead to a bad contact between the foot and the ground. If the foot is not anchored correctly on the ground a rotation on the stance hip will lead to a rotation of the stance leg instead of the desired rotation of the pelvis. To prevent those situations we add a supplementary torque $\tau_{contact}$ on the stance ankle that ensures a correct contact with the ground. We consider that the contact is correct if the sum of the vertical component of the ground reaction forces applied on the left side and right side of the foot stays higher than 10% of the sum of all the ground reaction forces on the stance foot. We use the same rule to ensure the balance between the front and the back. If needed the torque $\tau_{contact}$ is computed with the following equations:

$$\begin{cases} \tau_{contact}(x) = (0.1 - R_{F/B}) * K_{sag} \\ \tau_{contact}(z) = (0.1 - R_{L/R}) * K_{cor} \end{cases} \tag{3}$$

With $R_{F/B}$ (resp. $R_{L/R}$) being the ratio of either front or back forces (resp. left or right) that is under the 10% limit. K_{sag} and K_{cor} are gains similar to the ones used in the PD-controlers. Our experiments have shown that a value of 300 for both of them gives correct results.

6 Offline Optimization

The goal of the offline optimization is to find the optimal parameters of our controller for a given scenario (target character's sagittal speed and liquid height). The considered parameters are the 51 key points for the trajectory of the following elements: pelvis, lower back joint (L1 vertebrae), stance ankle, swing ankle, stance knee, swing foot.

6.1 Objective function

During our optimization we use the following evaluation function:

$$f_{eval} = \sum_{t<k} (\eta f_{energ} + \beta f_{drag} + \gamma f_{acc}) * (1 + 0.1 * R_{ipm\_alt}) + f_{speed} + f_{balance} + f_{phase} \tag{4}$$

where k is the duration in seconds of the evaluation. This function can be separated in two parts. The first part corresponds to the sum and defines the characteristics of the motion that we want to observe once the optimization is complete. We use a weighted sum of three functions, each one defining a specific behavior:

- *Minimization of consumed energy* (f_{energ}). The goal of this function is to obtain a motion using the minimum energy possible. This property is measured by the sum of norm of the torques at every joint: $f_{energ} = \sum_i ||\tau_i||$
- *Minimization of the drag.* (f_{drag}). Using this function, the character tries to move out of the liquid and we prevent velocity spikes within the liquid. We evaluate this property by using the sum of the torques induced by the drag forces on the parent joint of the limb where the drag forces F_j are applied: $f_{energ} = \sum_i \sum_j ((P_j(x,y,z) - P_i(x,y,z)) \times F_j)$ with $P_j(x,y,z)$ the force application point and $P_i(x,y,z)$ the position of the parent joint.
- *Minimization of angular acceleration* (f_{acc}). The goal of this function is to obtain smooth motions. We estimate the smoothness from the sum of the square of the angular accelerations weigthed by the mass of the corresponding limb in the reference poses $\ddot{\theta}_{di}$ and in the resulting motion $\ddot{\theta}_i$. We use two coefficients to favor minimizing the accelerations of the actual motion: $f_{acc} = \sum_i M_i * (0.25 * \ddot{\theta}_{di}^{\,2} + 0.75 * \ddot{\theta}_i^2)$

The second part of the evaluation function consists of functions limiting the search space:

- *Penalization of IPM alteration* (R_{ipm_alt}). The IPM alteration system is of great influence on the resulting motion. Even with reference poses defined to walk backward our system could succeed in walking forward if asked. To prevent this kind of situations, we penalize simulations heavily relying on the IPM alterations. To do so, we use the ratio between the IPM alteration required to obtain a stable motion ($\Delta(x,z)$) and its maximum threshold $max(\Delta(x,z))$ (0.09 in our tests): $R_{ipm\_alt} = \frac{\Delta(x,z)}{max(\Delta(x,z))}$
- *Velocity tracking* (f_{speed}). The goal of this function is to eliminate the simulations where the convergence to the target velocity takes too long (possibly due to unsuitable reference trajectories). We use a high penalization value if the error on the velocity tracking is superior to 1% of the target velocity: if $||V - V_d|| > 0.01 * ||V_d||$ then $f_{speed} = 10^{10}$
- *Motion balance* ($f_{balance}$). With this function, we verify if we have a stable motion by refusing simulations using recovery steps. If the system uses at least one recovery step, a high penalization value is given: $f_{balance} = 10^{15}$
- *Phase limits* (f_{phase}). This function is here to prevent generating motions that could easily reach a phase ϕ of 1.0 or that end steps at really low phases which limit our control on the character. In those cases, a high penalization value is given: if $\phi > 0.95$ or $\phi < 0.7$ at the end of a step then $f_{phase} = 10^{10}$

6.2 Optimization strategy

Our fitness landscape presents many local minimums. As several physics-based animation systems [Geijtenbeek et al., 2012; Tan et al., 2011], we use a Covariance Matrix Adaptation (CMA) [Hansen, 2006] to explore it.

7 Implementation

The physics engine used is Open Dynamics Engine (ODE). The simulation step size is $5\times10^{-4}s$. The human model used is composed of 28 degrees of freedom, see [Coros et al., 2009] for more details. Collisions between the character's limbs are not considered. The gains of the PD-controller are kept constant through the simulation, and are the same as in the forward walking controller of [Coros et al., 2009]. Joint torques are limited to $|\tau| < 200Nm$ for the hips and knees and $|\tau| < 100Nm$ for the other joints. The simulations are performed with a single-thread implementation on a common laptop with 8GB RAM and a 2.5GHz i5 processor. The controller proposed in this section as been build using a liquid with characteristics similar to water: $\rho = 1000kg.m^{-3}$ and $\mu = 1$. To compute the drag on any body part, we use finite elements around $0.02cm \times 0.02cm$.

8 Results

All the results are best seen in the accompanying video. (link to the video).

8.1 Offline Optimization

We report here the impact of the three main criteria in the evaluation function:

- The *minimization of consumed energy* leads to steps keeping the swing foot near the ground in absence of water. When partially immersed, the character keeps the swing foot low but elevates it a bit higher to align the foot with the direction of the foot's velocity.
- The *minimization of the liquid drag* leads to motions keeping the foot outside the water as long as possible until the water height is too high to do so (the limit is a bit above 0.5m which correspond to the knee height). It is interesting to note that the character uses a lateral movement to get the foot out of the water as fast as possible but it does not strike as natural looking.
- The *minimization of angular accelerations* leads to smooth motions keeping the swing foot near the ground with any water height but which prevents the character to recover from external pushes.

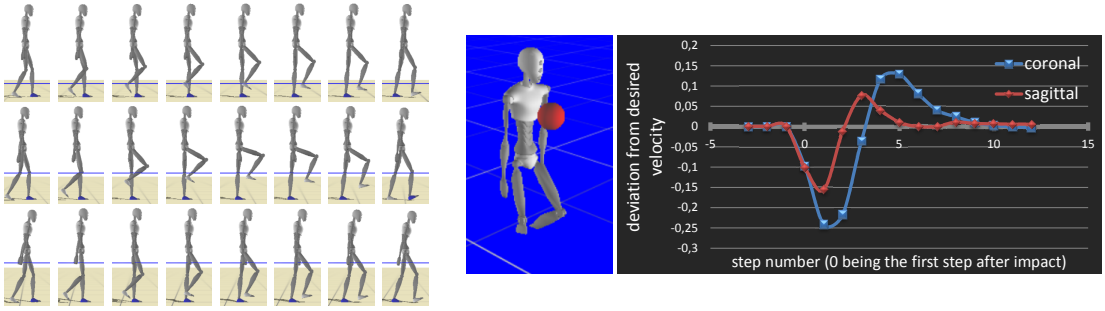

Fig. 3. Results obtained with our final controller. From top to bottom: character walking in 25cm, 50cm and 75cm of water. Right: character recovering from an external push, left: impact image, right: curves for the velocity deviation from the target velocity observed after impact. The character is moving at $0.6m.s^{-1}$ in all the presented situations with no coronal speed.

We experimented on the weights of the evaluation function, among which the two configurations ($\eta = 2$, $\beta = 8$, $\gamma = 1$) and (4,5,1). The first one of the two provoked the apparition of unnatural lateral displacements of the swing foot similar to our observations of the f_{drag} criteria. The second leaded to movements that would choose to stay underwater even for quite low water heights (under the knee). Finally, we have chosen to use the weights (3,6,1) to build our final controller. This combination leads to motions significantly influenced by the water height while keeping a natural looking movement. With these weights, we have created 10 reference controllers. We have two sets of reference controllers corresponding to the character's speeds $0.3m.s^{-1}$ and $0.7m.s^{-1}$. Each set contains one controller for each of the five following water heights: 0, 0.25, 0.5, 0.75 and 1 meter. Each optimization step has been realized with the following stages: 10 gait steps to stabilize the motion, then five seconds for the evaluation. The optimization of a set of five liquid heights for one speed took around 9 hours.

8.2 Experiments on the final controller

Our controller allows online modification of the character's target velocity. It is also possible to change the step width during movement. The controller is able to adapt to water height (figure 3) as well as changes to the liquid's physics properties (density and viscosity). Our system is able to withstand external forces simulated through solid balls of 5kg projected onto the character (figure 3 bottom). All the simulations respect the real-time condition with a minimum of 27 fps when the water reaches the hips.

Figure 3 illustrates some of the results. The first two rows show that the character adapts the height of the swing foot to reach the water height. The third row illustrates the case where the water height is too high and the character keeps the swing foot near the ground. The right images illustrate the character's recovery after an impact from a $5kg$ ball moving at $5.4m.s^{-1}$. As we can see, numerous steps are necessary to return to a stable velocity. The controller obtained by the presented optimization can precisely track any target sagittal speed from $0.2m.s^{-1}$ to $0.8m.s^{-1}$ under water height up to 1m and $1.0m.s^{-1}$ if the water height is lower than the knees. The following tests have been realized for a $0.6m.s^{-1}$ forward walk. The range of coronal speeds that can be achieved goes from $-0.25m.s^{-1}$ to $0.25m.s^{-1}$ under any water height up to 1m. The controller can adapt to liquids with a density up to 1.5 times the density of water. The character is robust to external pushes up to $5kg$ balls moving at $6m.s^{-1}$.

8.3 Discussions

In this work, we have focused on obtaining real-time simulations over having an extensive realism. Our controller could benefit from a more realistic water model that would still comply with the real-time constraint. Our system still uses a static threshold to detect the need for recovery steps instead of using an adaptive value depending on if the velocity learning has been completed or not. Using experimental results should be a correct solution to identify how the threshold should

be adapted. Our system presents some difficulties to reach the target coronal velocities especially if the desired variation is sudden. This is most likely caused by the fact that we need two learning velocity curves for the coronal axis. Adding a learning system with complex impact of the evolution of one curve on the other could be a possible solution.

9 Conclusion

We have presented a physics-based controller capable of real-time interactive simulation of walking motions for partially immersed bipeds. We have demonstrated online adaptation of the gait style to external conditions such as the liquid height and target character's velocity. Our controller permits the specification of new gait styles specific to walking motions in a liquid medium.

This project has been set up with the initialization of a research axis on motion control of virtual humans and will be extended during a Ph.D.. A paper has been submitted in july for the conference Motion In Games (MIG). I worked independently with the help of my supervisors during this project.

Our controller could be extended by implementing a more realistic liquid model which could consider the perturbations of the liquid caused by the character. Adding other mediums (e.g. snow-like mediums) or movement types (e.g. standing still, running...) with possible online transition between them are also interesting works we would like to investigate. Working on a non rigid model for the feet would improve the controller as well by bringing more advanced balance strategies and more realistic interactions between a virtual human and its environment.

Bibliography

Barela, A. M., Stolf, S. F., and Duarte, M. (2006). Biomechanical characteristics of adults walking in shallow water and on land. *Journal of Electromyography and Kinesiology*, 16(3):250–256.

Chevutschi, A., Alberty, M., Lensel, G., Pardessus, V., and Thevenon, A. (2009). Comparison of maximal and spontaneous speeds during walking on dry land and water. *Gait & posture*, 29(3):403–407.

Coros, S., Beaudoin, P., and van de Panne, M. (2009). Robust task-based control policies for physics-based characters. In *ACM Transactions on Graphics (TOG)*, volume 28, page 170. ACM.

Coros, S., Beaudoin, P., and van de Panne, M. (2010). Generalized biped walking control. In *ACM Transactions on Graphics (TOG)*, volume 29, page 130. ACM.

Geijtenbeek, T. and Pronost, N. (2012). Interactive character animation using simulated physics: A state-of-the-art review. In *Computer Graphics Forum*, volume 31, pages 2492–2515. Wiley Online Library.

Geijtenbeek, T., Pronost, N., and van der Stappen, A. F. (2012). Simple data-driven control for simulated bipeds. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 211–219. Eurographics Association.

Hansen, N. (2006). The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer.

Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., and Hirukawa, H. (2001). The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239–246. IEEE.

Lentine, M., Gretarsson, J. T., Schroeder, C., Robinson-Mosher, A., and Fedkiw, R. (2011). Creature control in a fluid environment. *Visualization and Computer Graphics, IEEE Transactions on*, 17(5):682–693.

Si, W., Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2014). Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics (TOG)*, 34(1):10.

Tan, J., Gu, Y., Turk, G., and Liu, C. K. (2011). Articulated swimming creatures. In *ACM Transactions on Graphics (TOG)*, volume 30, page 58. ACM.

Yang, P.-F., Laszlo, J., and Singh, K. (2004). Layered dynamic control for interactive character swimming. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 39–47. Eurographics Association.

Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: Simple biped locomotion control. In *ACM Transactions on Graphics (TOG)*, volume 26, page 105. ACM.

The following appendices are here to give to the reader the basis for the different concepts mentioned in this report. In particular, we propose one appendix for each one of the three following concepts: Joint-space motion control and PD-controllers, Inversed Pendulum model (IPM) and Inversed Kinematics with jacobian transpose. Those appendices are extracts from various papers previously mentioned. At the start of each appendix the source paper and the position of the extract within this paper will be mentioned. See the corresponding papers for a better understanding and to find the papers referenced inside the extracts.

A Joint-space motion control and PD-controllers

This appendix has been extracted from Geijtenbeek and Pronost [2012] page 8-9.

Joint-space motion control attempts to control physics-based characters by providing kinematic target trajectories, which are tracked using *local feedback controllers* that attempt to minimize the difference between the measured state and tar- get state. It has its origin in *control theory* and is the most common control method in industrial robotics [KSnL05].

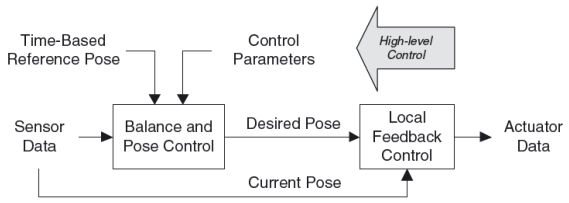

Fig. 4. Joint-space motion control

Figure 4 shows a schematic overview of joint-space motion control for physics-based character animation. Its main component performs *balance and pose control*, based on the current state, high-level control parameters, and (optionally) a time-based reference pose. It outputs a *desired kinematic state* for each joint, usually in the form of desired orientation and angular velocity. For each actuated joint, a *local feedback controller* computes a joint torque based on the difference between measured and target state.

The main appeal of this approach is its ease of implementation: all motion can be specified in the kinematic domain, based on kinematic insights, without the need for direct access to the underlying equations of motion. The downside of using local feedback controllers is that they operate individually, while coordinated motion is the product of all joints working together. The key challenge of joint-space motion control is in devising methods to compensate for this lack of coordination.

In the remainder of this section, we first describe techniques for local feedback control, followed by an overview of different approaches for generating kinematic targets.

3.1. Local feedback control

The purpose of a local feedback controller is to compute a torque that minimizes the difference between the current state and desired state of a single actuated joint. Physics-based characters generally contain rotational joints; kinematic targets are therefore mostly represented through joint orientation and angular velocity. However, feedback control strategies can be applied similarly to prismatic (sliding) joints.

3.1.1. Proportional-derivate control

The most widely used feedback control technique in joint- space motion control is called proportional-derivative control, or PD control. It computes a joint torque, τ , that is linearly proportional to the difference between the current state and a desired state. It takes into account both the difference in current orientation θ, and desired joint orientation θ_d , as well as the difference in current angular velocity θ, and desired angular velocity θ_d

$$\tau = k_p(\theta_d - \theta) + k_v(\dot{\theta_d} - \dot{\theta}) \tag{5}$$

In this equation, k_p and k_v are controller gains, and they regulate how responsive the controller is to deviations in position and velocity, respectively. Often, applications only specify a desired joint orientation and set desired angular velocity to zero ($\theta_d = 0$). A PD controller then becomes similar to a spring-damper system, where a spring generates a force to move to its rest position, θ_d . In such a system k_p defines the *spring gain* (or *spring constant*), and k_v the damping.

Parameter tuning. Finding correct values for k_p and k_v is not a straightforward task, and often requires manual tuning through *trial-and-error*. When controller gains are set too low, a joint may not be able to track its target and lag behind. When set too high, a joint may follow a target trajectory too rigidly and become stiff and unresponsive. The amount of rigidity is also referred to as *impedance* - in skilled human motion, impedance is usually low [TSR01, Hog90], while high impedance motion is regarded as robotic or stiff [NF02].

k_p and k_v must also be set in a proper relation to each other. A relatively high value for k_p may result in *overshoot* (meaning that a joint will oscillate around its desired position before reaching it), while a relatively high value of k_v may cause unnecessary slow convergence. When the relation between k_p and k_d is optimal (fastest convergence without overshoot), a controller is said to be *critically damped*. If the desired state and dynamics characteristics are fixed, the optimal relation between the two gain parameters is: $k_v = 2\ k_p$. For physics-based characters this relation is more complex, as both the desired state and the dynamics characteristics are subject to constant change. Still, this relation is often used as an estimate or a starting point for tuning.

B Inversed Pendulum Model

This appendix has been extracted from Coros et al. [2010] page 3.

Computing where to step and how to achieve that step is an important problem for walking skills. The steps we wish our controller to perform can vary widely in nature, ranging from making minimal shifts of foot position during idle stance to taking large steps during a fast walk or when receiving a large push.

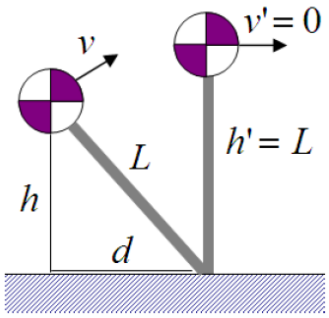

Fig. 5. Inverted pendulum model

We compute the desired stepping point, (X_d, Z_d), using an inverted pendulum model (IPM) in order to achieve a general solution that works across a wide range of body types. In particular, we build on an inverted pendulum model that assumes constant leg length [Pratt and Tedrake 2006] as shown in Figure 5. The analysis equates the sum of the potential and kinetic energy of the IPM at the current state, described by its current velocity v, its height, h, and distance, d, from the future

point of support, with that at the balanced rest state, i.e., $\frac{1}{2}mv^2 + mgh = \frac{1}{2}mv'^2 + mgh'$, where $v' = 0$ and $h' = L = \sqrt{h^2 + d^2}$. Solving this relation for d gives $d = v\sqrt{\frac{h}{g} + \frac{v^2}{4g^2}}$. The above model computes the desired value of d in order to reach zero velocity at the next step. Taking a shorter step will achieve a positive velocity while taking a larger step will achieve a negative velocity, i.e., walking backwards. Accordingly, we compute $d_0 = d - \alpha V_d$, where V_d is the magnitude of the desired velocity and α is a constant.

C Inversed Kinematics with jacobian transpose

This appendix has been extracted from Geijtenbeek et al. [2012] page 4.

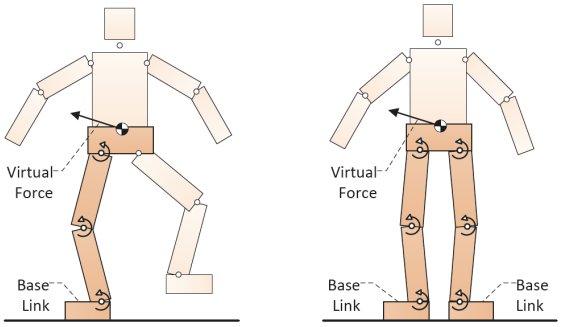

Fig. 6. Virtual forces applied to the center of mass, through a chain of linked bodies, for single stance (left) and dual stance (right).

The essence of this control method is that it enables control in Cartesian space for linked structures with redundant DOFs. Using the Jacobian Transpose, it is possible to compute the set of torques that emulate the effect of an external force or torque, applied to a specific body or a virtual point (such as the center-of-mass). Virtual forces and torques are applied to a *chain of linked bodies*, starting from a static base link (such as the stance foot) and moving to a target link (see Figure 6 for an illustration). The set of joint torques τ_F that emulate a virtual force F applied at point p corresponds to:

$$\tau = J_i^T F \tag{6}$$

where $J(p)$ is the *Jacobian* that represents the rate of change of a point p for each DOF i connecting the targeted chain of bodies. For a chain of bodies connected through k rotational DOFs, each row in $J(p)^T$ represents the rate of change of p with rotation a_i about DOF i:

$$J(p)^T = \begin{bmatrix} \frac{\partial p_x}{\partial \alpha_1} & \frac{\partial p_y}{\partial \alpha_1} & \frac{\partial p_z}{\partial \alpha_1} \\ . & . & . \\ \frac{\partial p_x}{\partial \alpha_k} & \frac{\partial p_y}{\partial \alpha_k} & \frac{\partial p_z}{\partial \alpha_k} \end{bmatrix} \tag{7}$$

For a rotational DOF i represented by normalized axis a_i and anchor position b_i (all defined in the world coordinate frame), the derivative $\frac{\partial p_x}{\partial \alpha_i}$ corresponds to the cross product between a_i and the relative position of p [Jaz10]. Hence, each row i of $J(p)^T$ corresponds to:

$$J_i^T(p) = \begin{bmatrix} \frac{\partial p_x}{\partial \alpha_i} & \frac{\partial p_y}{\partial \alpha_i} & \frac{\partial p_z}{\partial \alpha_i} \end{bmatrix} = (\alpha_i \times (p - p_i))^T \tag{8}$$

A virtual force F applied at point p can now be emulated by applying a torque tF;i to each DOF i that is part of the chain of bodies:

$$\tau = (\alpha_i \times (p - p_i))^T F \tag{9}$$

Similarly, it is possible to apply a virtual torque to a specific body at the end of a chain. If the orientation of the targeted body is represented using an *exponential map* [Gra98], $e \in \mathbb{R}^3$, then the rate of change of e with rotation a_i is identical to the normalized DOF axis a_i:

$$J_i^T(e) = \begin{bmatrix} \frac{\partial e_x}{\partial \alpha_i} & \frac{\partial e_y}{\partial \alpha_i} & \frac{\partial e_z}{\partial \alpha_i} \end{bmatrix} = a_i^T \tag{10}$$

Hence, a *virtual torque* T applied to the body at the end of the chain can be emulated by applying a torque τ_{Tj} to each DOF i that is part of the chain of bodies:

$$\tau_{Tj} = a_i^T T \tag{11}$$

Note that, since bipeds contain no links that are truly static, the effect of a virtual force or torque is always an approximation and can be determined only after simulation. That said, we have found this approximation to be sufficiently accurate to work well in practice.