SIGRAD 2014
M. Obaid, D. Sjölie, E. Sintorn and M. Fjeld (Editors)

PREDICTION OF PHYSICS SIMULATIONS FOR GRAPHICS AND ANIMATION

R. Dupre, V. Argyriou, D. Greenhill

Kingston University

Abstract

In this paper a novel approach for physics simulation prediction is proposed with applications in graphics rendering and multimedia. A prediction mechanism is introduced based on regression that aims to reduce the computational cost of simulations in a given scene by negating the need to perform physics calculations every frame. Novel features based on the energy of a scene over time are suggested for the training stage. Experiments were performed to evaluate the performance of the proposed prediction system, indicating that in cases where precision is not essential, regression tools can be utilized providing visually similar kinematics.

Physics, Performance, Correlation and regression analysis, Graphics rendering

Categories and Subject Descriptors (according to ACM CCS): - [Real-time rendering]: Visual Analytics—-

1. Introduction

Development of physics engines in computer games has had major advances in recent years. Also, the increased use of physics engines in other industries such as robotics and engineering, medicine, and mathematics [LLS09, HQZ12, SLM06] has led to the development of faster and more precise engines. In tandem the increase in computer performance has allowed and facilitated the development of such engines. Physics engines tend to focus on two main areas when it comes to simulating the physical world. The first is concerned with high precision, and aims to replicate the physical world as accurately as possible. The second is focused on the speed in which it can calculate these physical representations. An evaluation of free publicly available physics engines has been done [BB07], which highlighted that of the engines tested none were better at all experiments conducted and nearly all were better in one test than another.

Due to increasing physical complexity of scenes and the need for real-time rendering in combination with limitations of computer hardware it is important to find a balance between accurate representation and calculation time based on a set of parameters, such as distance and type of simulation.

In this paper we will demonstrate a novel dynamic approach in which the computation required to mimic physical aspects of a 3D environment can be reduced through the use of prediction techniques based on machine learning and Support Vector Regression. The proposed framework could learn the physic simulations during a game and dynamically switch to a prediction mode for a certain amount of time to reduce the overall complexity and computational workload. An analysis of the area will be followed by an overview of previous work. An outline of the basic concepts related to physics simulations and then the proposed methodology and results will be presented, leading to the final conclusions of this work.

2. Previous Work

Physics simulation plays an important role in many fields such as graphics rendering, multimedia, engineering, science, and education, allowing us to understand the laws of motion, matter, space and time. The ability to simulate physical behavior is critical in order to understand the complex physical world. This aids in improving design and realism in various industries such as multimedia and game applications, special effects and real-time rendering. Examples of industries that utilise visual simulation techniques where speed is a consideration include game production, and real time simulation environments such as flight simulators. In [Gou06] the fundamentals and basic methodologies for physics simulation can be found. A dynamic simulation approach for rigid bodies was proposed by Baraff in [Bar93] and in [Ega03] implementation techniques for real time rigid body simulation were suggested. Physics modeling for com-

puter games development and other multimedia applications was also analysed in [BM11, DMI11, SM12, RSH*13].

2.1. Regression and Support Vector Machines

Regression is the statistical process of analysing data sets to discover a relationship amongst its variables. Often used in the areas of forecasting and data analysis [WHP10], it provides us the ability to define and explore relationships between dependent and independent variables. Support Vector machines, originally designed as a classification algorithm, and the later devised Support Vector Machines for regression allows the presentation of a solution based on a small subset of training data.

3. Proposed Methodology for Physics Simulation Prediction

In this section, the problem of predicting physics mechanics in a given scene is re-formulated as a regression problem. Considering the advantages of Support Vector Regression (SVR), it was adopted as a regression tool in this work. Regression tries to estimate the relationship between a dependent variable (location or acceleration) and independent variables (a selected initial energy or force under a selected direction). This analysis allows the prediction of the physics dynamics in near future given the initial conditions for a short amount of time.

3.1. Regression problem formulation

SVR as a regression method was chosen for the proposed approach, because it features good generalization performance [SL06] that is essential for regression applications in combination with a polynomial kernel. SVR does not have a local minimum problem compared to other regression techniques such as neural networks. Additionally, SVR is not limited to the input space by using a linear kernel, but instead is operated in a arbitrary large feature space, since it is a kernel-based regression technique.

Let us assume that the training data is given as $\{(x_1,y_1),\ldots,(x_k,y_k)\} \subset X\times$, where k is the number of samples and X denotes the space of the input patterns (e.g. $X = R^d$), which might be a vector indicating the direction and magnitude of an applied force or equivalently an obtained acceleration and position). In the case of linear functions the regression equation $f(x)$ is defined as:

$$f(x) =< w,x > +b \tag{1}$$

where $w \in X, b \in R$, and $< \cdot,\cdot >$ denotes the dot product in X. The goal is to estimate a function $f(x)$ that satisfies the error from the difference between observed target y_i and the predicted value $f(x_i)$, with $i = 1\ldots k$, is disregarded as long as it is less than e. While at the same time it is as flat as possible by minimizing the norm $\|w\|^2 < w,w >$. Also, in order to avoid non feasible convex optimization problems some error is allowed analogously to the 'soft margin' loss function. Furthermore, slack variables ξ, ξ^* are introduced to cope with data points that lie outside the absolute e regions, and the following formulation is obtained.

$$minimise\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{k}(\xi_i + \xi_i^*) \tag{2}$$

$$subject\ to \begin{cases} y_i - < w,x_i > -b \le e + \xi_i \\ < w,x_i > +b - y_i \le e + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \tag{3}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than e are tolerated.

A Lagrange function is constructed from the objective function and the corresponding constraints, by introducing a dual set of variables. In order to solve more easily this optimization task the obtained dual optimization problem is defined as:

$$maximise\ to \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{k}(a_i - a_i^*)(a_j - a_j^*) < x_i,x_j > \\ -e\sum_{i=1}^{k}(a_i - a_i^*) + \sum_{i=1}^{k} y_i(a_i - a_i^*) \end{cases} \tag{4}$$

$$subject\ to \sum_{i=1}^{k}(a_i - a_i^*) = 0, a_i, a_i^* \in [0,C] \tag{5}$$

where $a_i, a_i^* \ge 0$ are the Lagrange multipliers. So, the objective function is obtained from the Support Vector expansion and can be written as follows:

$$f(x) = \sum_{i=1}^{k}(a_i - a_i^*) < x_i,x > +b \tag{6}$$

where the w can be described as a linear combination of the training pattern x as:

$$w = \sum_{i=1}^{k}(a_i - a_i^*)x_i \tag{7}$$

In order to apply a non-linear kernel to SVR in equations 5 and 6, the dot products $< x_i,x_j >$ for the linear kernel is replaced with a polynomial kernel defined as:

$$k^{pol}(x_i,x_j) = \beta_0 + \beta_1(x_i - x_j) + \ldots + \beta_p(x_i - x_j)^p + \varepsilon_i \tag{8}$$

The order of the polynomial was selected experimentally.

3.2. Feature selection and simulation

Based on the proposed methodology, a signal representation of a physical concept, such as acceleration, velocity or position over time, could be predicted using a few initial measurements and a set of training data. For example, instead of performing the complex physics simulations for an event's entire duration, only a few initial calculations over a very

short amount of time could be applied and the remainder of the simulation predicted using regression. This presents a huge computational saving at the cost of a minor decrease in visual accuracy. An example in a game environment; if the object or event that is affected is far away from the camera or is defined as being unimportant to accurately represent (i.e. not effecting the game play), prediction could provide a more efficient solution whilst maintaining the visualisation.

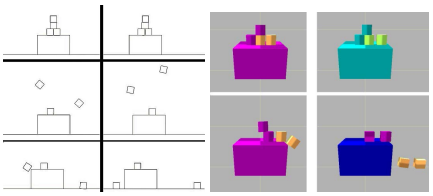

Figure 1: *An example of (left) 2D and (right) 3D simulated and predicted movements.*

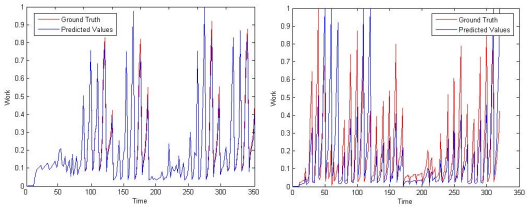

Figure 2: *Work over time for a (left) 2D and (right) 3D scene.*

Regarding the training dataset; regression is used to fully define a selected feature, in this case the Verlet Integrator acceleration, of an object in a scene over its duration. In order to obtain the position, we regard the velocity as the derivative of the position and the acceleration is the derivative of the velocity. This is repeated for a number of scenes each with forces of a variety of directions and magnitudes producing a model. For the prediction, a initial number of accelerations are simulated for each object in a scene. These initial values are then matched to one in the model using SVR and the polynomial kernel.

4. Results

In order to evaluate the performance of the proposed approach, experiments were performed using three physics engines. Box2D which performs constrained rigid body simulations, the Bullet open source physics engine featuring 3D collision detection, soft body dynamics, and rigid body dynamics; and the final, the NVIDIA PhysX engine which supports a number of game related features such as rigid and soft body dynamics as well as volumetric fluid simulation.

Both for the Box2D and Bullet engines, six scenes were designed and on each object an impulse force was applied. In the Box2D simulations, 36 force directions (sampling a circle every 10 degrees) and 10 different magnitudes were used to obtain the model. The same approach was used in the Bullet simulation, here the sampling was performed around a sphere. During the prediction stage the first 10 samples (locations of the scene objects) were calculated (less than 10%) and the remaining predicted. In figure 1 examples scenes are shown. In figure 2 the predicted and the calculated energy are plotted over time for two scenes demonstrating that predicted values closely follow those of the simulated. All scene results are shown in table 1 with the error varying from 1% to 15% based on the scene.

	Case1	Case2	Case3	Case4	Case5	Case6
2D	0.009	0.005	0.101	0.033	0.143	0.005
3D	0.017	0.009	0.102	0.045	0.153	0.009

Table 1: *Average percentage error based on measured difference between predicted and simulated values for 2D and 3D scenes for all scenes.*

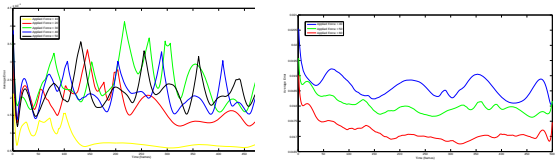

Figure 3: *The Sequential and Explosion Model's average error per frame between training data and the regression output (per Force).*

To further test this concept, two experiments were set up using the NVIDIA PhysX engine to visualise familiar scenarios in a multimedia environment. The first a simulated explosion, the second a waterfall. For each scenario, training data was collected and used as the basis for the prediction to reduce the computational complexity of these effects.

For the explosion scene, two models were created. In the first; samples of different forces were taken from around a sphere ('sequential' model). The second created training scenes in which 100 cubes (particles) have a force applied in various strengths and directions with each cube being recorded individually ('explosion' model). The accuracy of a scene is measured by the Euclidean distance between the predicted track of each particle and the original simulated track, normalised across all the particle data in the scenes.

Error	Sequential	Explosion	Both	Water
Force 1	0.013	0.011	0.010	0.023
Force 2	0.017	0.015	0.013	0.029
Force 3	0.022	0.024	0.017	0.036

Table 2: *Sequential, Explosion, Both and Water Models average error per force during testing.*

Figure 3 shows the average error that each model has over time against its original training track due to the regression. For testing; five explosions were simulated each with 3 forces for a 10 second period using 100 cubes (particles). The models were then used to predict each cubes track in 3D

space, using the initial 10% of simulated values to match a predefined trajectory. Matching was done by calculating the function curve that matched closest with the initial simulation samples during the training. Visualisations of the results are shown in figures 4 and 5. Table 2 demonstrates the accuracy of each model in relation to a simulated track, as well as the robustness of the method across differing scenes.

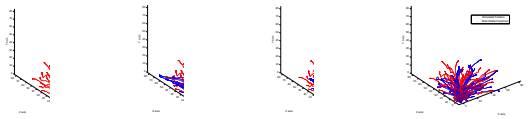

Figure 4: *Visualisations of a symmetric explosion (a) ground truth trajectories and the predicted ones for each model, (b) 'explosion', (c) 'sequential' and (d) both, (red is the ground truth and blue the estimated trajectories).*

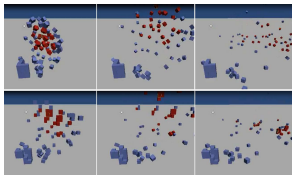

Figure 5: *Visualisations of an asymmetric explosion (top) ground truth trajectories and (bottom) the predicted ones for the 'sequential' model, (the red cubes indicate the applied forces).*

The same experiments were performed in the waterfall scene. A single model was created during the training comprising of five various waterfall scenes each with three different forces. Tests were carried out on 4 test waterfall scenes each with 3 forces. Examples of the outputted prediction verses the original simulation are scene in figure 6. The visualisations demonstrate that we can get a very close approximation of the original tracks though the use of the proposed methodology, with the overall average error shown in table 2. In our test environment we predict the movement of the particles in a scene for 90% of the time, this represents a considerable computational saving.

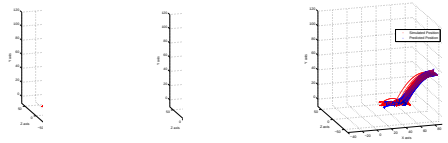

Figure 6: *Visualisations of real (red) and predicted (blue) trajectories for a test scene, three different forces.*

5. Conclusion

In this paper, a novel approach to predict simulated physics was proposed based on regression. Also, a proposed framework can operate dynamically and learn the prediction models during the game play. From the obtained results it can be observed that for cases that accuracy is not essential such as distant events or special effects that humans cannot distinguish the differences significant reduction in the overall required computational cost can be achieved.

References

[Bar93] BARAFF D.: Non-penetrating rigid body simulation. *State of the art reports*, May (1993). 1

[BB07] BOEING A., BRÄUNL T.: Evaluation of real-time physics simulation systems. *Proceedings of the 5th international conference on . . . 1*, 212 (2007), 281–288. 1

[BM11] BOGDAN P., MARCULESCU R.: A fractional calculus approach to modeling fractal dynamic games. *IEEE Conference on Decision and Control and European Control Conference* (Dec. 2011), 255–260. 2

[DMI11] DELGADO-MATA C., IBÂŤNEZ J.: Adaptive Physics for Game-Balancing in Video-Games for Social Interaction. *2011 International Conference on Technologies and Applications of Artificial Intelligence* (Nov. 2011), 254–259. 2

[Ega03] EGAN K.: Techniques for Real-Time Rigid Body Simulation. 1

[Gou06] GOULD, HARVEY; TOBOCHNIK, JAN; CHRISTIAN W.: *Introduction to Computer Simulation Methods: Application to Physical Systems*. Addison-Wesley, 2006. 1

[HQZ12] HU W., QU Z., ZHANG X.: A New Approach of Mechanics Simulation Based on Game Engine. *Computational Sciences and . . .* (2012). 1

[LLS09] LUO F., LIU C., SUN Z.: Intelligent Vehicle Simulation and Debugging Environment Based on Physics Engine. *2009 International Asia Conference on Informatics in Control, Automation and Robotics* (Feb. 2009), 329–333. 1

[RSH*13] ROENNAU A., SUTTER F., HEPPNER G., OBERLAENDER J., DILLMANN R.: Evaluation of physics engines for robotic simulations with a special focus on the dynamics of walking robots. *2013 16th International Conference on Advanced Robotics (ICAR)* (Nov. 2013), 1–7. 2

[SL06] SAKHANENKO N., LUGER G.: Shock physics data reconstruction using support vector regression. *. . . of Modern Physics C 17*, 9 (2006), 1313–1325. 2

[SLM06] SERVIN M., LACOURSIERE C., MELIN N.: Interactive simulation of elastic deformable materials. *Proceedings of SIGRAD Conference* (2006), 22–32. 1

[SM12] SILVA D. F., MACIEL A.: A comparative study of physics engines for modeling soft tissue deformation. *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)* (Oct. 2012), 1–7. 2

[WHP10] WU J., HUANG L., PAN X.: A novel bayesian additive regression trees ensemble model based on linear regression and nonlinear regression for torrential rain forecasting. *2010 Third International Joint Conference on Computational Science and Optimization* (2010), 466–470. 2