Using Motion Levels Of Detail in the Fast Multipole Method for Simulation of Large Particle Systems

Seyed Naser RAZAVI
Department of computer engineering, Iran University of Science and Technology
Tehran, Iran

Nicolas GAUD
Multi-agent systems and applications group, Laboratoire Systèmes et Transports, UTBM
Belfort, France

Abderrafiâa KOUKAM
Multi-agent systems and applications group, Laboratoire Systèmes et Transports, UTBM
Belfort, France

Naser MOZAYANI
Department of computer engineering, Iran University of Science and Technology
Tehran, Iran

razavi@iust.ac.ir, nicolas.gaud@utbm.fr, abder.koukam@utbm.fr, mozayani@iust.ac.ir

ABSTRACT

This article introduces a novel approach to increase the performances of *N*-body simulations. In an *N*-body simulation, we wish to evaluate all pairwise interactions between *N* bodies or particles. The direct computation of all pairwise interactions requires O(N^2) time, which is clearly prohibitive for a very large *N*. Our approach combines the Fast Multipole Method (FMM) coming from computational physics with motion levels of detail from computer graphics. The main goal is to speed up the execution of the *N*-body simulations while controlling the precision of the associated approximation, a natural trade-off between accuracy and efficiency common in the field of simulation. At each simulation cycle, the motion levels of detail are generated automatically and the appropriate ones are chosen adaptively to reduce computational costs. The new approach follows the overall structure of the FMM. However, clusters are approximated using their Center of Mass (CoM) in force computations. A similarity measure is used to decide which clusters can be approximated without any significant loss in the accuracy of the simulation. The proposed approach is tested for Coulombic system, in which *N* charges induce potentials to each other. The preliminary results show a significant complexity reduction without any remarkable loss in the visual appearance of the simulation, indicating the potential use of the proposed model in the simulation of a wide range of *N*-Body systems.

Key words – *Dynamics Simplification, Fast Multipole Method, Motion levels of Detail, Multi-Agent Based Simulation*

1. INTRODUCTION

Nowadays, Thanks to the increasing power of modern computers, simulation plays a very important role in the study of large complex systems and has attracted many researchers from a growing number of diverse areas including sociology, biology, physics, chemistry, ecology, economy, etc. Simulation of a large complex system often requires computing all the pairwise interactions between particles in the system. Assuming *N* particles, direct methods require O(N^2) computations which is clearly prohibitive for large *N*. The challenge of efficiently carrying out the related calculations is generally known as the *N-body* problem.

The Fast Multipole Method (FMM), proposed by Rokhlin and Greengard [1,2,3], can be regarded as one of the most successful attempts to reduce the O(N^2) time complexity required in the *N*-body simulations. Also, it has been identified as one the ten most important algorithmic contributions in the past century [4]. Depending on the distribution of particles in the computational domain, FMM evaluates the pairwise interactions in O(*N* log *N*) or even O(*N*) which is a remarkable improvement over the O(N^2) time required by direct methods.

The FMM is based on the idea that well-separated or far-away groups of particles can be considered as one particle. So, it imposes a hierarchical spatial partitioning structure on the computational domain to determine the near-field and the far-field for each particle or group of particles. Based on our recent work [5], we think that this hierarchical structure can be exploited to further reduce the overall computational cost. As the "no free lunch" theory suggests, this complexity reduction introduces a new source of inaccuracy in the simulation which is tolerable in many applications where we are more interested in the overall behavior of the system as a whole instead of the individual behavior of the constituent particles. The new approach presented here combines the FMM with motion levels of detail from computer graphics to improve simulation speed in the *N*-body simulations while controlling the desired level of accuracy.

In fact, the FMM algorithm clusters particles into hierarchical groups based on their relative positions in the domain, but it deals with each particle in the same group separately, computing a different potential and force for each particle in the group resulting in different behaviors for different particles in the same group. However, particle systems generally exhibit a high level spatial coherence, meaning nearby particles behave in approximately the same way. Therefore, the main idea is to compute only one approximated behavior for a group and then making all particles in the group to follow the same behavior. This is achieved by replacing particles inside a cluster by one weighted particle and approximating the motion models of those particles in the force computations during FMM.

To perform automatic dynamics simplification, a mechanism is needed to generate a hierarchy of motion models and also to switch between different levels in the hierarchy. Here, we have decided to use the same *hierarchical physically-based subdivision* scheme in the FMM. This way we can take advantage of the approximate computations used in the FMM to reduce the complexity beside our automatic dynamics simplification. However, during the force computations, some parts of the FMM tree (quadtree in 2D and octtree in 3D) are pruned and replaced by only one node containing a single weighted particle. Particles which are located in the pruned areas follow the same behavior computed for the weighted particle. This way, a lot of computations can be saved.

The pruning described above can result in a huge amount of inaccuracy. Thus, to keep the error below a desired level specified by the user, a mechanism is used to determine which parts of the tree can be pruned safely without any remarkable loss in the visual appearance of the simulation. This important decision that specifies which parts of the tree can be pruned is made at each simulation cycle. Therefore a specific region in the computational domain may be simulated in a higher resolution in the current cycle and it may be simulated in a lower resolution at later cycles because of the change in the distribution of its particles. It is very important to note that this transition between different levels of detail should be very smooth so that it does not cause a noticeable change in the flow of the simulation.

Therefore, at each simulation step the tree is updated based on changing simulation requirements as defined by the user or by the nature of simulation. Appropriate motion levels of detail are adaptively generated based on this subdivision given the requirements for different regions of the simulation and a desired execution time for each step. We have implemented a prototype system than can automatically generate simplified motion models, select appropriate models, and switch between them seamlessly. Then, the proposed framework is applied to the dynamic simulation of a large ensemble of particles in a Coulombic system which is a common example for *N*-body systems. The preliminary results are very promising, indicating a significant reduction in the complexity of the simulation while maintaining its correctness, and thus the potential to generalize the framework to the dynamic simulation of more systems.

Although there are existing simulation acceleration techniques such as [6,7,8,9], there is no known algorithm for the *automatic dynamics simplification* of complex physical or biological systems especially when all the pairwise interactions between particles are needed. In this paper, we limit the scope of our investigation to particle systems as a first step toward the design of automatic dynamics simplification. Particle systems are commonly used in computer graphics, physically based modeling, and animation of natural phenomena and group behavior. Additionally, they are often the building blocks of highly complex dynamical systems. Therefore, we hope the results of this paper can lead to the generalization of dynamics simplification for a broader class of dynamical systems (e.g. pedestrians and crowds simulation).

The rest of this paper is organized as follows: section 0 gives a brief review of some related works regarding simulation acceleration techniques and motion levels of detail. Section 0 presents a quick review of the FMM which is used to solve our target application. Section 4 describes the proposed framework, including the automatic generation of motion levels of detail and the mechanism used to select appropriate models and to switch between them adaptively. Some experimental results are given in section 5. Finally, section 6 concludes with some future research guidelines and perspectives.

2. LITERATURE REVIEW

In this section, various attempts related to level of details done in the field of interactive computer graphics are summarized. To achieve *dynamic realism* in computer graphics, often the polygonal geometry of small or distant portions of the model is simplified to reduce the rendering cost without a significant loss in the visual content of the scene. Therefore, the goal of polygonal simplification in rendering is to reduce the complexity of a polygonal model to a level that can be rendered at interactive rates. In an offline preprocessing step, multiple versions of each object are created at progressively coarser *levels of detail*, or LODs. Once the LODs have been created and stored for every object in the model, complexity can be regulated at run-time by choosing for each frame which LOD will represent each object. As an object grows more and more distant, the system switches to coarser and coarser LODs. This type of simplification is called geometrical simplification or graphical levels of detail. It is important to distinct it from simulation levels of detail.

Simulation levels of detail or motion levels of detail techniques have been proposed for reducing the computational cost of the dynamics simulation of the character motion. Motion models can be generated from pre-recorded motion sequences, procedurally approaches, kinematics, or based on dynamics computation. Some of the earlier human motion models in computer animation exploited this concept implicitly by using procedurally generated motion, simplified dynamics and control algorithms, off-line motion mapping, or motion play-back [10,11,12,13,14].

Carlson and Hodgins applied simulation LOD techniques to a graphical environment populated with multiple, physically simulated one-legged robots [6]. They demonstrated that a group of characters that dynamically switches between LODs can sufficiently replicate the performance of a fully simulated group. Although in this work, the generation of simulation LODs, switching and selection are designed by hand but their experimental results are indicative of the potential of automatic simplification of general dynamical systems.

By using the space-time constraint dynamics formulation, Popovic and Witkin introduced a motion transformation technique that preserves the essential properties of animated character motion with drastically less number of degrees of freedom [9]. Multon, et al. suggested a series of simplified walking models for mobilizing on complex terrain, as well as how and when the transition takes place [15].

Other types of simulation acceleration techniques have also been investigated to reduce the total computational simulation costs for a large, complex dynamical system. For example, Chenney *et al.* proposed view-dependent culling of dynamic systems to speed up the computation of dynamics by ignoring what is not visible to the viewer [7], similar to view culling. As another example, Faloutsos *et al.* developed a system that accomplishes complex tasks by evaluating multiple controllers automatically and selecting an appropriate sequence [16].

Additional graphics researchers are investigating ways to simplify physical simulations to reduce computation costs. Grzeszczuk *et al.* developed a technique that uses neural network approximations to emulate physically simulated characters' equations of motion [8]. After the neural network is trained to sufficiently model the original system, it can produce motions more efficiently than a full dynamic simulation. O'Sullivan and Dingliana investigate the opportunities to replace simulated particles with simplified counterparts when imperceptible to the viewer based on the idea that inaccurate dynamics are less noticeable in the peripheral vision and when the movements are complex [17]. O'Brien *et al.* describe a method to automatically simplify particle simulations through clustering into spatially localized groups [18].

It is very important to notice that the practitioners of interactive computer graphics are concerned less with accurately simulating the physics of a system than with finding better and faster ways to approximate the results of such a simulation. This is very different from a situation in which accuracy of simulation is paramount. However, with careful attention, it is possible to apply the same ideas to these kinds of applications to reduce the time complexity while preserving accuracy. This is a major feature distinguishing this work from those in the field of computer graphics. Therefore, the work presented in this paper may be considered as a link connecting the rich body of literature on real-time computer graphics with those in N-body simulations.

3. REVIEW OF THE FMM

Given N source densities $\{\varphi_i\}$ located at N source points $\{x_i\}$, we wish to compute the potential $\{q_i\}$ at N target points $\{y_i\}$ induced by a kernel G using the following relation:

$$q_j = \sum_{i=1}^{N} G(x_i, y_j)\phi(x_i), j = 1, \dots, N.$$

As mentioned earlier, direct implementation of this summation result in an O(N^2) algorithm, while for a large class of kernels, FMM computes the same interactions in O(N) time. However, FMM is an approximate algorithm, in the sense that the summation is not computed exactly. But, the good news is that the error can be bounded from above and the constant factor in the time complexity of the FMM is directly related to the accuracy of the approximation.

In our implementation, we have used the single layer Laplacian kernel ϕ which arises from solving the Coulombic system of charged particles. Given a point charge of unit strength at point x_i in the complex plane, then for any point y_j also in the complex plane with $x_i \neq y_j$, the potential at point y_j due to the charge of x_i is given by $G(x_i, y_j) = -\log\|y_j - x_i\|$. In the FMM context it is convenient to use $G(x, y) = Re(\log(z_y - z_x))$ where z_x and z_y are complex numbers corresponding to x (source) and y (target) points on the plane. The main idea of FMM is to represent the potentials of a set of source densities using the *multipole expansion* and *local expansion* at places far away from these sources. Assuming that the source densities are located inside a disk centered at z_c with radius r, then for all z outside the disk with radius R ($R > r$), the potential at z from the source densities can be represented using a set of coefficients $\{a_k, 0 \le k \le p\}$ where

$$q(z) = a_0 \log(z - z_c) + \sum_{k=1}^{p} \frac{a_k}{(z - z_c)^k} + O\left(\frac{r^p}{R^p}\right) \quad (1)$$

This is called multipole expansion and represents the contribution from the source densities inside the disk centered at z_c with radius r on the far-field of that disk. Similarly, the contribution from source points in the far-field on any target point z inside that disk is called local expansion and can be represented again using a set of coefficients $\{c_k, 0 \le k \le p\}$ where

$$q(z) = \sum_{k=1}^{p} c_k (z - z_c)^k + O\left(\frac{r^p}{R^p}\right) \quad (2)$$

In both expansions, the *truncation* number p is usually a small constant determining from the desired accuracy of the result. For the definitions of the coefficients $\{a_k\}$ and $\{c_k\}$, the reader may refer to [2].

The above representations are used by FMM in a recursive manner. This is achieved by a hierarchical partitioning of the computational domain. At the first step of the FMM, which is called *space partitioning*, the computational domain, a box large enough to contain all source and target points, is hierarchically partitioned into a tree structure (a quadtree in 2D or an octtree in 3D). The tree is constructed so that the leaves contain no more than a prespecified number of points, say s. This parameter is usually called the *clustering size*. Figure 1 shows an example distribution of particles and the corresponding quadtree in 2D.

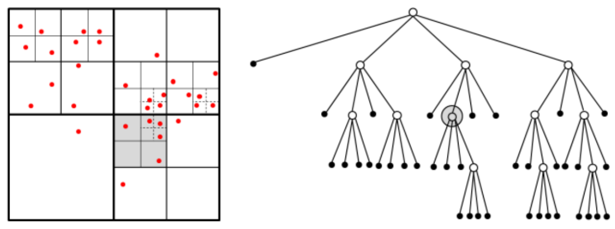

Figure 1 A 2D particle distribution (left) and its corresponding quadtree (right).

After the tree construction, FMM performs two different passes of the tree with different goals: one *upward pass* and one *downward pass*. During the upward pass, the tree is traversed from bottom to the top in a postorder manner. The goal of the upward pass is to compute the multipole expansion for each box in the tree. For each leaf box, the multipole expansion is computed directly from its source densities using (1), while the multipole expansions for nonleaf boxes are computed using an operator which is called M2M translation (see Figure 2b).

During the downward pass, the tree is traversed in preorder to compute the local expansions. For each box B, the local expansion is the sum of two parts: first, the local-to-local transformation collects the local expansion of B's parent (the contribution from the sources in all boxes which are not adjacent to B's parent) as shown in Figure 2d, and second, the multipole-to-local transformation collects the multipole expansions of the boxes which are the children of the neighbors of *B*'s parent but are not adjacent to *B* (these boxes compose the *interaction list* of *B*) as shown in Figure 2c. Therefore, the some of these two parts encodes all the contribution from the sources in the boxes which are not adjacent to *B* itself.

At the final step of the FMM, which is called *final summation*, for each box the near interaction evaluated by iterating over all the source points in the neighborhood of the target box is combined with the far interaction which is evaluated using local expansion at this box to obtain the potential of each target point inside that box.

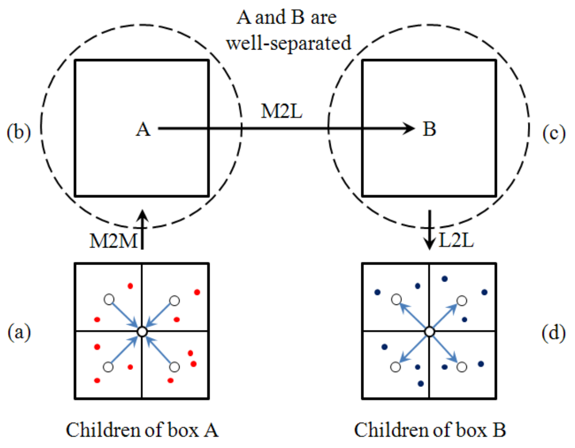

Figure 2 The Fast Multipole Algorithm mechanisms: (a) after spatial decomposition, (b) the child boxes use the Multipole to Multipole translation to shift their multipole expansion to the center of the parent box, (c) using the Multipole to Local translation, well-separated boxes interact by creating a local expansion at the center of box *B* due to box *A*, (d) the children of box *B* feel the potential of box *A* by using the Local to Local translation to shift the parent's local expansion.

In addition to these expansions (multipole and local) that can be used for an efficient evaluation, FMM uses three different types of translation operators to translate between these expansions. The combined effect of these expansions and translations make an O(*N*) complexity algorithm possible. These translation operators are listed here:

M2M: the *multipole to multipole* translation transforms the multipole expansions of a box's children to its own multipole expansion.

M2L: *the multipole to local* translation transforms the multipole expansions of a box to the local expansion of another non-adjacent box.

L2L: the *local to local* translation transforms the local expansion of a box's parent to its own local expansion.

For the derivation of these translations and a detailed discussion on complexity analysis, error bounds and implementation details, the reader may refer to [2] and [19].

4. INTEGRATING MOTION LEVELS OF DETAIL INTO THE FMM

This section presents the new method for automatic dynamics simplification with application in the simulation of large dynamical systems. As the new method follows the overall structure of the FMM, the focus of this section is mainly on the differences rather than the similarities. The discussion starts by presenting the details about computing the *Center of Mass* (CoM) particle for each box or group of boxes, and then follows by introducing the criteria used in the new algorithm to determine those parts of the tree that can be pruned to simplify calculations.

4.1 Approximating motion models

For each box (a leaf box or an internal box in the FMM tree), dynamics simplification is achieved by approximating its particles dynamics using the particle dynamics of its center of mass. Given a box *B* consisting of *k* particles, the corresponding approximated motion model is computed using the following procedure:

1) Compute the position P_{COM} and velocity V_{COM} of the center of mass particle using:

$$P_{COM} = \frac{\sum_{i=1}^{k} m_i P_i}{\sum_{i=1}^{k} m_i}$$

$$V_{COM} = \frac{\sum_{i=1}^{k} m_i V_i}{\sum_{i=1}^{k} m_i}$$

   Where m_i, P_i and V_i representing the charge, position and velocity of the *i*th particle inside *B*. Using the standard computations in the FMM, first compute the potential of the CoM particle for box *B* and then update its velocity and position accordingly.

2) Apply the same results computed for the COM particle to all particles inside *B*.

4.2 Pruning the FMM tree

After the construction of FMM tree, some parts of the tree are pruned to speed up the simulation process. This pruning is performed based on the past behaviors of the particles in each box. For a given box, if the behaviors of the particles are more or less the same, then the particles inside that box can be replaced with the CoM particle of that box. So, a mechanism is needed to decide for which boxes the dynamics computation can be simplified. Here, these boxes are called *simplified boxes*. Applying a good mechanism is a key factor determining the overall success or failure of the automatic dynamics simplification.

In this work, two similarity measures are used to determine the candidate boxes for simplification: the *speed ratio* and the *velocity vector angle ratio*. These computations differ for a leaf box and an internal box:

- If *B* is a leaf box, these measures are computed from the velocities of its particles. That is, the relative speed of its particles and also their relative angle should be within certain ratios.

- Otherwise, if *B* has children, the similarity measures for *B* are computed by considering the COM particles of its children. Again the relative speed of the COM particles and their relative angle are constrained to be within certain ratios.

The above computations are done during the upward pass of the FMM algorithm. If box *B* satisfies these two conditions, then all of its particles are replaced by its weighted center of mass particle which is computed directly from its particles or indirectly from the center of mass of its children. After this simplification, the new algorithm continues as the classical

FMM. However, when the potentials have been computed for every particle (a real particle or a weighted center of mass particle), the computed potentials for a CoM particle are applied to all of the particles inside the corresponding simplified box. A snapshot of the developed system for 200 particles is given in Figure 3. Because of dynamics simplification, the number of particles is reduced from 200 to 63 in this system resulting in a considerable reduction in the execution time.

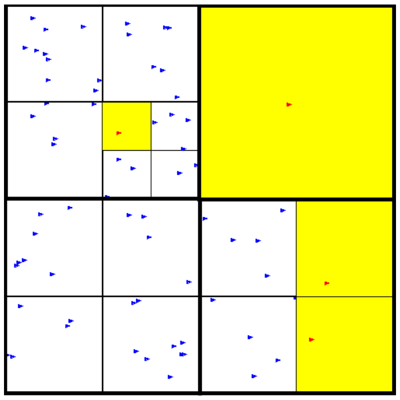

Figure 3 A snapshot of the developed system with 200 agents. Each colored box represents a box whose particles dynamics are simplified using dynamics of its CoM particle (the red particle in it).

4.3 Real-time simulation

During the simulation, it is possible to monitor the execution time of the last few steps and compare it to the target execution time defined by the user. If the execution speed is too slow, the system performs more simplifications by enlarging the *maximum cluster size*. This is achieved by allowing greater boxes (those that are located at higher levels in the hierarchical tree) to be merged. The inverse occurs when the speed is more than necessary. This is achieved by introducing a new parameter called *simplification level*. For a given simplification level, only boxes below that level are allowed to be merged. A small value for this parameter means that boxes at higher levels are allowed to be merged and a large value means that only small boxes at lower levels are allowed to be merged. Therefore, each time we need a faster execution, the simplification level is decreased and this allows larger boxes in higher levels to be merged. Increasing this parameter does the inverse.

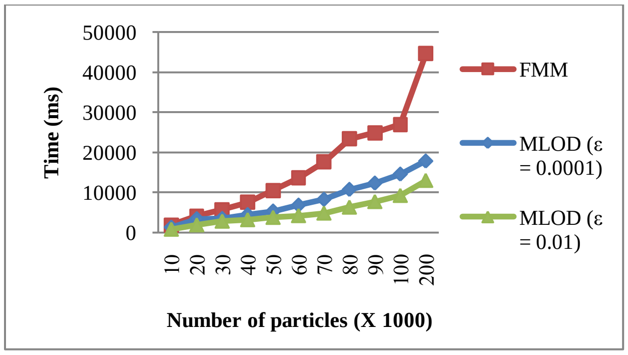

Figure 4 Comparing execution time of the original FMM to the new algorithm.

In summary, the proposed framework has several parameters which enable us to control both the error and the execution time of the simulation. These parameters are: the speed ratio and the velocity vector angle ratio, the maximum number of particles in each box and the maximum cluster size or the simplification level. The set of these parameters gives us the potential to apply our framework for real-time applications as well. Future works are already planned to test this real-time capability of the proposed framework.

5. EXPERIMENTAL RESULTS

To test the proposed method, we have developed a prototype system in java with the ability to generate motion levels of details, select the appropriate ones and switch between them. This section presents the results of the new method tested on the particles in a Coulombic system, which is a common example for *N*-body systems. All experiments in this section were conducted on a desktop computer configured with one 2.5GHz Quad-Core Intel Pentium processors and 4GB RAM running a Windows 7 Professional x32 Edition. Our application was developed in Java and the Java VM used to execute the tests was configured with 1GB memory.

5.1 Simulation of a large dynamical system

The goal of this experiment is to compare the execution time of the proposed method to the original FMM, while trying to keep the error below the user-specified level. For each experiment, the appropriate values for similarity measures are determined by try and error. Also, all the results are computed by averaging the execution times over 100 consecutive runs. To keep the results comparable, we have fixed the truncation number p to 10 and the clustering size s to 100 in all experiments. The results are shown in Figure 4.

As it can be seen in Figure 4, the experimental results confirm the theoretical expectations, i.e., more computational time can be saved if the algorithm is allowed to be run less accurately. Of course, it is possible to achieve more efficiency by relaxing accuracy requirements more. This trade-off between accuracy and efficiency enables us to cope with very large and complex systems which would be intractable otherwise.

For a better understanding of how adding motion levels of detail can affect efficiency, we can introduce the *effective number of particles* for the proposed method. For a given amount of execution time, this parameter specifies that how much particles can be processed by the two methods. For example, consider the case in which number of particles is equal to 200000 for the new method. In this case, the execution time is about 13 seconds which is equal to the required time by the original FMM when there are approximately 60000 particles. Therefore, for 200000 particles, the effective number of particles is 60000 which is less than one-third of the original size. Table 1 reports the effective number of particles in the new FMM algorithm for various numbers of particles. All the given numbers in this table are computed approximately using interpolation.

Table 1 Effective number of particles in the proposed method

Particle No. (× 1000)	Effective Number of Particles (× 1000)	
	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-4}$
10	3.5	5
20	10	15
30	12	17
40	15	21.5
50	16	32
60	20	35
70	23	44
80	33	50
90	40	52
100	49	56
200	60	72

6. CONCLUSIONS

This paper presents a new framework for simulation of large particle systems by combining the well-known fast multipole method from computational physics with the concept of motion levels of detail from computer graphics. The main goal is to reduce the computational complexity of the FMM by adding automatic dynamics simplification to it and hence to speed up the simulation of large *N*-body. Dynamics simplification is achieved by first replacing all particles inside a box with one weighted center of mass particle and then applying the potential computed for the CoM particle to all particles in that box.

This recursive dynamics simplification can be performed for any box at any level. This provides us a hierarchy of approximated motion models which are updated at each simulation cycle. These motion levels of detail enable us to adjust the level of simulation dynamically and hence to trade accuracy for efficiency. The results given in section 5 are very .promising indicating the potential use of the method to more dynamical systems. However, this work is a first step toward the design of a general multilevel simulation framework and can be extended in several ways:

- Investigating other possible methods or heuristics which can be served as the similarity measure.
- Designing new experiments to gain a better understanding of performance of the system and giving a more complete complexity and error analysis by studying the effects of main parameters such as truncation number on the complexity and the error of the new method..
- Extending the proposed method for real-time situations (see section 4.3).

REFERENCES

[1] V. Rokhlin, "Rapid solution of integral equations of classical potential theory," *Journal of computational physics*, pp. 187-207, 1983.

[2] Leslie Greengard and Vladimir Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325-348, 1987.

[3] Leslie Greengard and Vladimir Rokhlin, "Rapid evaluation of potential fields in three dimensions," in *Lecture notes in mathematics*, vol. 1360, Berlin, 1988, pp. 121-141.

[4] J. Dongarra and F. Sullivan, "The top ten algorithms of the century," vol. 2, no. 1, pp. 22-23, 2000.

[5] N. Razavi, N. Gaud, N. Mozayani, and A. Koukam, "Multi-agent based simulations using fast multipole method: application to large scale simulations of flocking dynamical systems," *Artificial Intelligence Review*, vol. 35, no. 1, pp. 53-72, 2011.

[6] D. Carlson and J. Hodgins, "Simulation levels of detail for real-time animation," in *Graphic Interface 1997*, 1997.

[7] S. Chenney and D. Forsyth, "View-dependent culling of dynamic systems in virtual environments," in *ACM Symposium on Interactive 3D Graphics*, 1997.

[8] R. Grzeszczuk, D. Terzopoulos, and G. Hinton, "Neuroanimator: Fast neural network emulation and control of physics-based models," in *SIGGRAPH*, 1998, pp. 9-29.

[9] Z. Popovic and A. Witkin, "Physically based motion transformation," in *SIGGRAPH 1999*, 1999, pp. 11-20.

[10] A. Brudlerlin and T. Calvert, "Goal-directed, dynamic animation of human walking," in *Computer Graphics (Proc. of SIGGRAPH'89)*, 1989, pp. 233-242.

[11] A. Brudlerlin and T. Calvert, "Knowledge-driven, interactive animation of human running," in *Graphics Interface*, 1996, pp. 213-221.

[12] M. Girard and A. Maciejewski, "Computational modeling for computer animation of legged figures," in *Computer Graphics (Proc. of SIGGRAPH)*, vol. 19, 1985, pp. 263-270.

[13] J. Granieri, J. Crabtree, and N. Badler, "Production and playback of human figure motion for 3d virtual environments," in *VRAIS*, 1995, pp. 127-135.

[14] k. Perlin, "Real time responsive animation with personality," in *IEEE Transactions on Visualization and Computer Graphics*, 1995, pp. 5-15.

[15] F. Multon, B. Valton, B. Jouin, and R. Cozot, "Motion levels of detail for real-time virtual worlds," in *ASTC-VR'99*, 1999.

[16] P. Faloutsos, M. van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *SIGGRAPH 2001*, 2001, pp. 251-260.

[17] C. O'Sullivan and J. Dingliana, "Collisions and perception," in *ACM Transactions on Graphics*, 2001.

[18] D. O'Brien, S. Fisher, and M. Lin, "Automatic simplification of particle system dynamics," in *Computer Animation*, p. 2001.

[19] H Cheng, Leslie Greengard, and Vladimir Rokhlin, "A fast adaptive multipole algorithm in three dimensions," *Journal of Computational Physics*, vol. 155, pp. 468-498, 1999.