EG UK Theory and Practice of Computer Graphics (2009)
Wen Tang, John Collomosse (Editors)

# An Edge-based Approach to Adaptively Refining a Mesh for Cloth Deformation

T. J. R. Simnett  S. D. Laycock  A. M. Day

School of Computing Sciences, University of East Anglia
Norwich, NR4 7TJ, UK
{t.simnett|s.laycock|a.day}@uea.ac.uk

**Abstract**

*Simulating cloth in real-time is a challenging endeavour due to the number of triangles necessary to depict the potentially frequent changes in curvature, in combination with the physics calculations which model the deformations. To alleviate the costs, adaptive methods are often employed to refine the mesh in areas of high curvature, however, they do not often consider a decimation or coarsening of areas which were refined previously. In addition to this, the triangulation and consistency checks required to maintain a continuous mesh can be prohibitively time consuming when attempting to simulate larger pieces of cloth. In this paper we present an efficient edge-based approach to adaptively refine and coarsen a dynamic mesh, with the aim to exploit the varied nature of cloth by trading the level of detail in flat parts for increased detail in the curved regions of the cloth. An edge-based approach enables fast incremental refinement and coarsening, whereby only two triangles need updating on each split or join of an edge. The criteria for refinement includes curvature, edge length and edge collisions. Simple collision detection is performed allowing interactions between the cloth and the other objects in the environment.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

## 1. Introduction

The role of cloth in computer graphics has increased in the last decade, especially with character animation for films and games. Many methods have been used to simulate cloth and clothing, often focusing on the visual appearance and the physical properties. A high degree of fidelity has already been achieved for off-line applications, however, simulating realistic cloth remains an expensive endeavour, even when considering recent advances in computer hardware. One can sacrifice detail to achieve an interactive cloth simulation with a coarse mesh relatively easily, but to simulate detailed cloth interactively or even just to accelerate off-line computations additional techniques must be employed.

Interactions between the cloth and other surfaces in the environment complicates its shape, typically resulting in a multitude of curves, ripples and planar sections. Regions of high curvature require more points and polygons to ensure a reasonably accurate approximation which increases the computation time. Conversely, largely planar regions require less detail and can be simulated relatively cheaply. Adaptive meshes aim to exploit the varied nature of cloth by trading the level of detail in flat parts for increased detail in the curved regions of the cloth, where it is required. The mesh is refined typically by subdividing the cloth surface comprising of triangles or quads.

In this paper, we present an edge-based adaptive refinement approach applied to a triangular mesh where the subdivision is the result of splitting the edges in the mesh. The criteria for edge splitting are important and depend on the application. However, curvature, edge length and edge collisions are suitable criteria for cloth modelling. Triangle subdivision schemes such as bisection, 1-to-4 splits and $\sqrt{3}$-refinement have been used previously [VL03] [LV05]. A common difficulty concerning triangle subdivision is that one must frequently either build a conforming mesh that bridges successive layers or deal with the resulting T-junctions that occur in the mesh. The proposed edge-based strategy ensures the mesh is always conforming and that it



DOI: 10.2312/LocalChapterEvents/TPCG/TPCG09/077-084

can be used directly. The triangles present in the mesh are subdivided into two, three or four further triangles based on the status of its three edges. Internally each triangle can be in one of eight states, these states do not allow T-junctions to form. The refinement is recursively defined whereby each edge has two child edges; triangles serve as containers to internal edges and triangles on the next level. In order to preserve the regularity of triangulation, we only allow further subdivision of edges and therefore subdivision of internal triangles when the complete 1-to-4 split has taken place. Edges function as length preserving springs but are also very useful for facilitating cross-springs for bending forces. A split edge does not contribute any force to the connected particles, it becomes inactive and its child edges automatically take over the work of calculating edge length and cross spring forces.

The paper is organised as follows: Section 2 discusses previous work, focusing on adaptive simulations. Section 3 details our contribution, Section 4 showcases the results and Section 5 concludes the paper.

## 2. Previous Work

Cloth simulation has been researched for both real-time and offline applications with many different methods developed for refining a mesh which strive for a more computationally efficient implementation. An interesting generic method for incremental mesh adaptation which supports any triangle refinement rule such as 1-to-4 split or $\sqrt{3}$-refinement was presented in [VL03]. Their method was based on a hierarchy of semi-regular meshes, requiring a conforming mesh to be constructed that bridges different layers of the mesh using triangles from the highest available resolution. Direct updates to the conforming mesh alleviated the need to rebuild it. It was used in three applications, a physically based cloth simulation, real-time terrain visualisation and procedural modelling. View dependent criteria for refinement, which includes distance, are important for terrain visualisation, where subdivision should only occur for the area of the mesh in view. However, with cloth simulations the shape is of most importance and therefore we will also employ curvature, as in other adaptive cloth simulations [VB05] [LV05].

Thomaszewski et al. presented a method to model consistent bending using co-rotational subdivision finite elements in order to correctly simulate the folding and buckling behaviour of cloth [TWS06]. They compared the compression of a real fabric cylinder with simulated ones. Their method produced a very good likeness to the real one. In comparision using a standard simple bending model, the fold patterns were different to the real sample, also the patterns completely changed using a finer mesh with the simple model. Their method is not suitable for real-time simulations but achieves excellent realism. Volkov and Li found that fine wrinkles observed in the non-adaptive simulation were missing from their adaptive one, these fine wrinkles were attributed to buckling behaviour which cannot be detected by the curvature based criterion [VL03]. Mujahid et al. mention the possible use of stretching as a refinement criteria, though they did not link this to its use with buckling [MKM*04]. We propose to simulate buckling by an additional edge splitting criterion, this is detailed in Section 3.5.

The use of adaptive methods to simulate cloth introduces problems which are particularly prevalent when the mesh deforms. Care is needed to preserve the cloth's physical properties to avoid visually distracting artefacts around the subdivision seams. Villard and Borouchaki presented an adaptive method to allow the mechanical system to behave without any constraint related to a uniform mesh. New points are added to the mesh based on local curvature at that point. Four elements are subdivided into sixteen around the point creating eight new active nodes and eight virtual nodes. This becomes more complicated at successive refinements, some virtual nodes become active and the number of new nodes varies in order to preserve the warp/weft structure. The adaptive simulation and uniform simulation over a sphere produce almost identical results as shown by a superimposed image of the two final meshes together, nonetheless the simulation time is improved by as much as six times [VB05]. However, a major limitation of the approach is that the refinement could not be reversed.

Etzmuß et al. presented a particle system that adaptively generates new particles whenever they are necessary in order for collisions to be correctly handled. This allowed them to perform fast and accurate simulations with coarse meshes. Virtual points applied forces to the mesh by springs but were not integrated fully into the simulation as their position was calculated from edge collision and was only valid for one time step [EEHS00]. Previously, non-active points had been used for correct collision handling [HH98]; savings were realised by not having to simulate those points. The non-active points are inserted half way along edges in a quad mesh and also in the centre of each quad. The constraint that a uniform mesh can only bend along predefined lines was effectively removed and the cloth could bend as required. The mesh was able to drape over edges of objects accurately although there is noticeable stretching as a result of the non-active points being moved to resolve collisions. They concluded that either edge rest lengths must be adapted or the non-active points must influence the simulation to overcome the stretching; however, they did not resolve this. Sifakis et al. presented a hybrid simulation of deformable solids, which did not suffer from the above problems although it was not accomplished in real-time. Their framework allowed embedding arbitrary sample points into a mesh for handling collisions, plasticity and fracture without the need for complex remeshing. Hard bindings constrain a particle to the barycentric coordinates of their location, having no degrees of freedom they redistribute forces applied to their parents and are used to deal with T-junctions. A soft binding connects a simulated particle to a hard binding target location enabling two-

way interaction between the particle-based system and the mesh-based framework [SSIF07].

Hardware has moved towards the increased use of multi-core processors where data sharing becomes complicated, since simultaneous data writes must not be allowed. Mujahid et al. effectively used OpenMP to implement a cloth simulation with adaptive refinement and coarsening in parallel. Load balancing was an important feature to this and it improved simulation times by up to 32% when running on 8 cores even without considering cache coherency [MKM*04]. Their simulation is slower for early iterations, where a higher density mesh is used and is then coarsened in flat areas as the simulation progresses. It would be advantageous to simulation times if a coarse mesh could be used from the onset, this strengthens the case for using collisions as a refinement criteria.

Zhang and Yuen presented a fast cloth draping simulation using multilevel meshes, by dividing the process into several phases. The whole mesh is subdivided after each phase, thereby increasing the detail of the mesh. After each phase the mesh is closer to its final position [ZY01]. This is only suitable for speeding up cloth in draping situations where the cloth has a final resting position. The time chosen to enter the next phase is crucial; once the coarser mesh has reached its balanced position, the final shape cannot be improved noticeably by further subdivision. Advancing phases too quickly will cause the performance gains of this method to be lost, so a balance must be struck between quality versus speed.

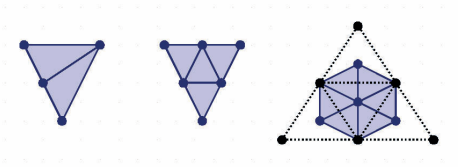

**Figure 1:** *Various subdivision Schemes, Left: Bisection, Middle: 1-4 Split, Right: $\sqrt{3}$-refinement.*

The $\sqrt{3}$-refinement rule is where vertices are inserted into triangle face centres; this results in up to six child triangles with each child having two parents. Figure 1 shows this and two more refinement rules, child triangles are coloured blue. Li and Volkov applied it to clothes on an animated character [LV05]. A conforming mesh was extracted from a hierarchy, with adaptation taking 87ms and the simulation taking 1.2 seconds on average per step for 12k triangle adaptive meshes. Their adaptive simulation used more triangles than the non-adaptive one but was of higher quality.

An interesting alternative to adaptive simulations, but highly related, are subdivision surfaces where the surface of a mesh is subdivided, typically to increase the detail for rendering and lighting without imposing significant costs on storage and animation for large meshes. This can therefore be applied to coarse cloth simulations to improve the visual quality without increasing the cost of the simulation. Curved PN-triangles are a good example of a generic method that can be performed in hardware and is applied to triangles [VPBM01]. Three sided cubic Bézier patches form each curved PN-triangle, coefficients are calculated such that neighbouring patches match together perfectly without the need for adjacency information. Each patch is specified by only the triangle's three vertices and their normals. Lorenz and Döllner presented a dynamic refinement strategy using geometry shaders on the GPU with an incremental multi-pass scheme [BS05]. Subdivision surfaces may be of limited use for cloth which is not often considered by itself. Typically, the simulation involves complex interactions with other objects which occur as a result of collisions. Therefore, one possible difficulty is that the rendered surface may intersect the object even if the coarse cloth mesh does not.

Large adaptive meshes can present additional challenges to remain efficient. The memory architecture of the computer must be taken into account where out-of-core storage is needed. Large terrain models in [VL03] exploited the subdivision connectivity of the adaptive mesh by storing related data in each data block, their hierarchical storage layout out performed linear storage by more than an order of magnitude. Typically only each 100th data request resulted in disk access.

## 3. Method

In this section we first present a brief overview of our cloth simulation followed by our main contributions. We explain the steps and processes that enable edge refinement and coarsening. We explore the critera used to trigger adaption and finish by detailing our state based retriangulation approach.

### 3.1. Cloth Simulation

To simulate the deformations of the cloth as forces are excerted upon it a mass-spring system is employed. The cloth is constructed from vertices, defined by point masses, which are connected by springs. Mass-spring systems are popular for their ease of implementation and for real-time performance [DB99]. However, they can suffer from stability and accuracy problems. Parameters must be carefully tailored to the situation and often by trial and error. Provot was the first to explore the use of mass systems for Cloth, in particular dealing with the unrealistic behaviour which resulted from over stretching of springs. To overcome this, Provot utilised a dynamic inverse procedure to correct the lengths of super-elongated springs [Pro95], we perform one iteration of this at each simulation step. We use Verlet integration to numerically integrate Newton's equations of motion for the cloth vertices (masses), where velocity is derived from the current

and previous positions. Forces are internal to the cloth, generated by Hooke's law type linear springs.

The following list details the steps of the simulation combining the cloth simulation, collision detection and adaptive mesh effectively together. These steps must be performed at run-time:

1. Spring Force Calculation and Accumulation
2. Numerical Integration
3. Edge Length Constraining and Refinement
4. Point-Object Collision
5. Edge-Object Collision and Refinement
6. Surface Normal Calculation
7. Curvature Based Edge Refinement
8. Re-triangulation

### 3.2. Edge Refinement

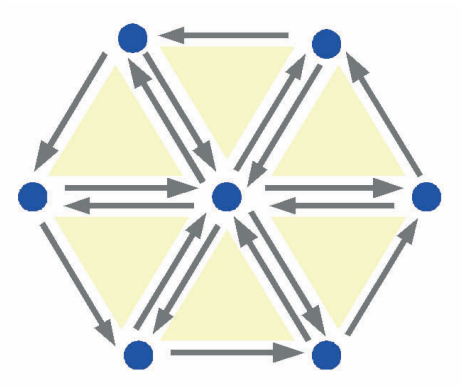

**Figure 2:** *Edges (grey), Vertices (blue) and Triangles (yellow) for lowest detail level constructed from a small base mesh. Notice that there is exactly three times the number of edges than triangles, two opposite edges between two adjacent triangles form edge pairs.*

A Base Mesh of triangles is loaded and used to construct the Vertices, Edges and Triangles that make up the Adaptive Mesh. Memory for successive levels of the adaptive mesh can be preallocated before the simulation begins, when this memory is exceeded dynamic allocation occurs at runtime if required for further refinement. Dynamic allocation imposes only a singularly occuring time penalty for each new vertex, edge or triangle as the memory is not released until the simulation has ended. Edges are defined in an anti-clockwise order around each triangle where there is exactly three times the number of edges as triangles. Our approach relies on a large amount of connectivity information for the mesh, it is the key to the ease of refinement and coarsening. The connectively information is built into the new edges and triangles as the mesh is adapted. This permits a very efficient algorithm for refining and coarsening which overshadows the expense of maintaining the connectivity.

Pairs of edges are connected together between two adjacent triangles with references to each other, they share two vertices and have opposite directions. Figure 2 illustrates the topology for a small mesh of six triangles, eighteen edges and seven vertices. For each of the edge pairs only a single reference to one of them needs to be stored. Operations such as splitting or joining a stored edge are automatically mirrored in the other edge. Edges also store references to the adjacent triangle on their 'left', as viewed from above, to aid in triangulation and a reference to the third vertex on that triangle to be used for cross-springs. Cross-springs are defined over each pair of edges using the 3rd vertex references, see Figure 3. Edges that lie on the boundary of the mesh do not have any connected oppositely directed edge, therefore there is no pair to form a cross-spring and splitting or joining only requires one triangle to be retriangulated. Both edges and cross-springs are linear springs; they apply forces to vertices that try to restore their lengths. The rest lengths are stored as part of the edges, they are calculated from material coordinates which ensures correct responses as the cloth deforms.

New vertices are added to the centre of edges as they are split with each vertex being assigned the average position, previous position and material coordinates of the two adjacent vertices. An approach which incorporates multiple refinement criteria is prone to excessive work due to potentially rapid changes in refinement and coarsening caused by conflicting criteria. For example, an edge collision could cause an edge to split but the curvature criteria could potentially immediately rejoin the edges, this could also happen with the edge length criteria. To overcome this, a time limit is imposed which starts after a split such that a rejoin event is prevented for a number of simulation steps after the split event.

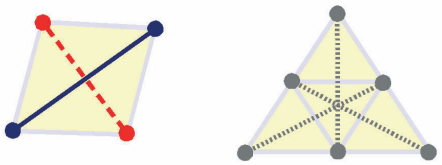

**Figure 3:** *Left: An edge pair (blue) and its cross-spring (red), Right: Cross-springs shown for the centre triangle's three edges.*

### 3.3. Edge Coarsening

In a dynamic simulation of cloth, it is important to be able to reverse the refinement in the mesh in regions where the detail is no longer required. We perform coarsening as the opposite to edge refinement, two child edges are rejoined together such that they and the centre vertex become inactive and their parent edge becomes active once again. In our

incremental approach, a split edge is prevented from being rejoined if its child edges are themselves refined.

### 3.4. Curvature Driven Refinement and Coarsening

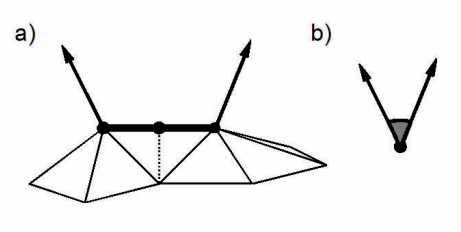


**Figure 4:** *a) Normals for two vertices shown for the highlighted edge on part of a mesh. b) Curvature is defined locally as the angle between the two normals, calculated by the dot product.*

We define curvature locally to each edge by the angle between the two surface normals of the two vertices at the end of each, this is shown in Figure 4. When the angle is greater than an adjustable amount, the edge is split; conversely when it is less than another amount the edge can be rejoined. Smooth normals are calculated for each vertex from the average of normals to adjoining triangles, each vertex stores a small list of adjacent triangles for this purpose. The vertex normals are also used for rendering using OpenGL's smooth shading model.

### 3.5. Length Driven Refinement

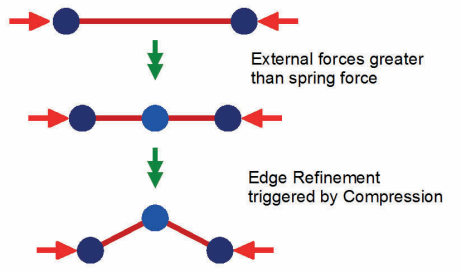


**Figure 5:** *Buckling behaviour simulated by Edge Compression triggering local refinement, the new centre vertex of the split edge is free to move and it generates wrinkles in the cloth.*

As previously mentioned, we aim to simulate buckling behaviour in the cloth by using edge length as a criterion for refinement. We constrain edge lengths to overcome the cloth stretching unrealistically, but when it is compressed real cloth will tend to wrinkle and fold. A coarse mesh cannot do this easily, therefore refinement is needed to allow the adaptive mesh to buckle. When two vertices are pushed together by stronger forces than the edge spring can oppose, the edge becomes in compression. We define a minimum percentage of length, any edge with less than this length is split, see Figure 5. Now that the edge is replaced by two child edges, the cloth can start to buckle enabling more visually pleasing folds to be generated.

### 3.6. Collision Driven Refinement

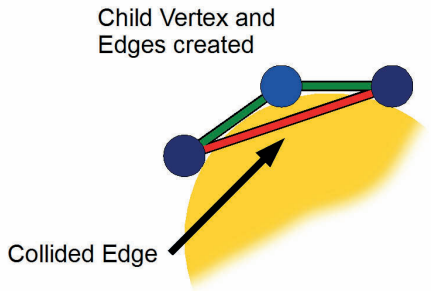


**Figure 6:** *Collision between an edge and object (sphere) triggering local refinement, the new centre vertex is projected onto the object's surface.*

We perform simple collision detection between the cloth and objects such as spheres and cylinders, we do not consider self-collision. Point-object collision detection and response when combined with a coarse mesh and a relatively small object will result in the cloth falling through the object or the object sticking out of the cloth in places. A solution to this is to perform full polygon-polygon collision detection and response but this is much more computationally expensive. If the fast point-object detection is to be used, the mesh must have enough vertices required to resolve the collision correctly. We therefore want to cause refinement in the adaptive mesh around the area of the collision, even if the curvature is not sufficient to cause edge splitting. Firstly, Point-Object collision is performed between the Mesh and objects, anticipating that many polygon-object collisions will be resolved by this step. Vertices in collision are projected out onto the object's surface and can smoothly slide over it. Collision detection between the edges of the mesh and the objects is performed. The edges deemed in collision are split; the new centre vertex is then projected out of the surface as before, see Figure 6. At first glance this method appears to suffer from stretching, the sum of two child edges lengths will be greater than their parent's length. However, as the new vertex is fully integrated into the cloth simulation successive iterations will quickly restore the edge lengths and the effect is not noticeable.

### 3.7. State Based Re-triangulation

A triangle's internal configuration is represented by its state, there are eight main states as shown in Figure 7. State 0 rep-

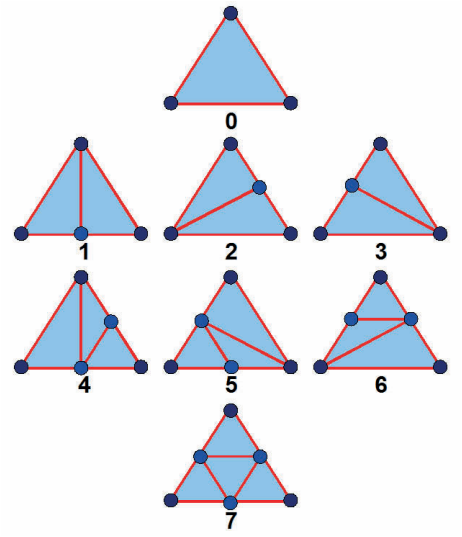


**Figure 7:** *Triangle states showing allowed internal configurations. Edge splits cause retriangulation based on state transitions.*

resents a triangle with no subdivision whilst State 7 represents a triangle that has completed a full 1-to-4 split and may have many internal triangles at higher levels. Retriangulation of a triangle involves calculating a state transition, given a triangle's current state and the status of its three edges, its new state can be determined quickly. The triangle is then reconfigured into that state and update tags are reset to false (previously set by the edge splits and joins). Retriangulation is a costly operation, it is not ideal to immediately perform it on the two adjacent triangles when an edge is refined or coarsened, this will lead to situations where triangles are retriangulated up to three times instead of once. Therefore, the retriangulation is delayed until the final simulation step, at a point where all edges are up to date and no furthur changes are known to be required before rendering. Another consideration is that the regularity of the mesh must be preserved as it is subdivided. To support this, before an edge is refined it must be validated to ensure that the split is permitted. To enable this, child triangles are assigned to State -1, this is of identical configuration to State 0 but further subdivision is prevented. Once a complete 1-to-4 split occurs and the triangle has reached State 7, all four child triangles have their states set from State -1 to State 0 allowing for subsequent refinement. Our approach is fast and incremental, a mininum number of vertices and triangles are used to refine the mesh at each step.

## 4. Results

The approach was implemented in C++ and OpenGL. The results presented here were performed on a 2.66 Ghz Intel Core i7 920 processor (only 1 core was used). The timings in the figures are for the whole mesh at specified levels, these represent the absolute worst case for our adaptive mesh, since in practice only parts of the mesh will refine to a particular level during a simulation step. Figure 8 shows the Simulation and Rendering time of a base mesh consisting of thirty two triangles and the subsequent levels leading to a refined mesh of over 131K triangles. Our results prior to level 5 are very fast, enabling interactive frame rates. The simulation part is the most computationally expensive, however, this could be approximated exploiting coherence between time steps to achieve interactive rates on meshes capable of refining to higher levels. Our Edge-based approach was designed for local incremental change; nevertheless refinement for the whole mesh is fast, refining the mesh from level 0 (32 triangles) to level 5 (32k triangles) required approximately 8.5ms. See Figure 9 for other transition times.

| Level | Tri. | Vert. | Simulation | Render |
|---|---|---|---|---|
| 0 | 32 | 25 | 0.0109 ms | 0.0033 ms |
| 1 | 128 | 81 | 0.0547 ms | 0.0132 ms |
| 2 | 512 | 193 | 0.3131 ms | 0.0545 ms |
| 3 | 2048 | 417 | 1.3895 ms | 0.2326 ms |
| 4 | 8192 | 865 | 5.6428 ms | 1.0036 ms |
| 5 | 32768 | 1761 | 24.0329 ms | 3.9151 ms |
| 6 | 131072 | 3553 | 91.0788 ms | 13.5329 ms |

**Figure 8:** *Maximum triangle and vertex counts for the specified mesh level, corresponding simulation and rendering times are given.*

| Mesh Transition | Increase Level | Decrease Level |
|---|---|---|
| Level 0 - 1 | 0.0164 ms | 0.0096 ms |
| Level 1 - 2 | 0.0745 ms | 0.0452 ms |
| Level 2 - 3 | 0.3396 ms | 0.2044 ms |
| Level 3 - 4 | 1.3720 ms | 0.7932 ms |
| Level 4 - 5 | 6.7124 ms | 4.5453 ms |
| Level 5 - 6 | 27.0744 ms | 16.7605 ms |
| **Level: 0 - 5** | **8.5149 ms** | **5.5977 ms** |

**Figure 9:** *Refinement and coarsening times for a whole mesh for the specified level transitions.*

Performance of the adaptive mesh is directly related to the objects it collides with, in particular the relative size of the object compared to the size of the triangles in the cloth. A high level of detail is required in the mesh to lay on a small sphere, conversely the coarse base mesh may be enough to lay upon a large sphere. The three criteria for edge splitting allows the detail to automatically change to match with the object that it is in collision with. This is illustrated in Figure 10, the cloth is shaded using a colour scale corresponding to the subdivision level of each triangle. Also the insert shows that our approach generates similar defor-

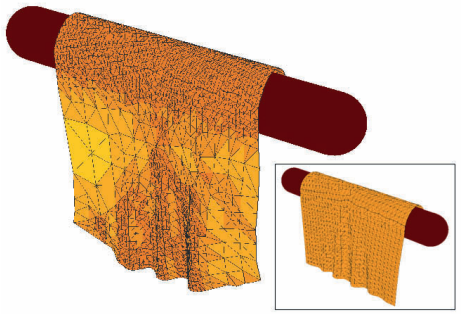

**Figure 10:** *Cloth after falling onto a cylindrical beam. The image is rendered to show the subdivision levels, darker colours represent the higher detail levels. Bottom-Right Insert: Uniform mesh shown for comparision.*

mations compared to a uniform mesh. Our adaptive mesh performs particularly well in that situation, where regions in contact with the beam are heavily refined. As the cloth slides off the beam, the mesh can then coarsen once again. Figure 11 shows the time per simulation step, for 7 seconds of the simulation with the maximum level set to four. Initially the time per step is very fast, increasing as rapid refinement is triggered by the collision. The time per step does not exceed two milliseconds while the cloth is sliding over the beam and then it gradually recovers as the cloth becomes relatively flat again. The draping of the cloth on a sphere can been seen in Figure 12. The results are illustrated in the accompanying video at the following url: http://www.urbanmodellinggroup.co.uk/ClothSim.wmv

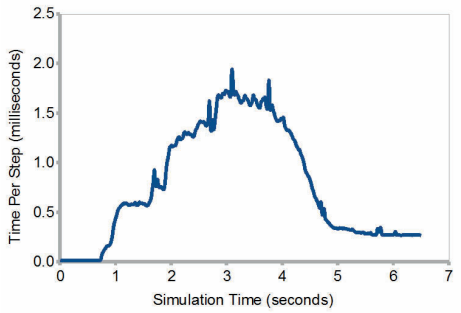


**Figure 11:** *Time Per Step for an adaptive simulation of the Cloth falling onto and sliding off of a cylindrical beam, aproximately 70 simulation steps are performed per second in real-time. These times exclude collision detection.*

## 5. Conclusion

In this paper, we have presented a fast edge-based adaptive mesh for cloth simulations which allows refinement and coarsening dynamically based on edge length, curvature and

**Figure 12:** *Cloth after falling onto a sphere, rendered with smooth lighting.*

collisions. The method is suitable for most situations where a dynamic mesh is needed and can be extended for different splitting criteria easily. Triangles are defined as being in one of eight states and transitions can be calculated in a few inexpensive operations. In the future we aim to apply our technique to the simulation of clothing on virtual human characters. The use of more sophisticated collision techniques will be required to enable complex interactions to be calculated efficiently. Also, it would be prudent to investigate the parallelisation of this approach to allow for more detailed cloth while keeping it Real-Time.

## References

[BS05] BOUBEKEUR T., SCHLICK C.: Generic mesh refinement on GPU. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2005), ACM New York, NY, USA, pp. 99–104.

[DB99] DESBRUN M., BARR A.: Interactive animation of structured deformable objects. In *In Graphics Interface* (1999).

[EEHS00] ETZMUSS O., EBERHARDT B., HAUTH M., STRASSER W.: Collision adaptive particle systems. In *Proceedings Pacific Graphics 2000* (2000), vol. 4.

[HH98] HOWLETT P., HEWITT W.: Mass-Spring Simulation using Adaptive Non-Active Points. In *Computer Graphics Forum* (1998), vol. 17, Blackwell Synergy, pp. 345–353.

[LV05] LI L., VOLKOV V.: Cloth animation with adaptively refined meshes. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-*

*Volume 38* (2005), Australian Computer Society, Inc. Darlinghurst, Australia, Australia, pp. 107–113.

[MKM*04] MUJAHID A., KAKUSHO K., MINOH M., NAKASHIMA Y., MORI S., TOMITA S.: Simulating Realistic Force and Shape of Virtual Cloth with Adaptive Meshes and Its Parallel Implementation in OpenMP. In *Proceedings of international conference on parallel and distributed computing and networks (PDCN2004)* (2004), pp. 386–91.

[Pro95] PROVOT X.: Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. In *GRAPHICS INTERFACE* (1995), CANADIAN INFORMATION PROCESSING SOCIETY, pp. 147–147.

[SSIF07] SIFAKIS E., SHINAR T., IRVING G., FEDKIW R.: Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), Eurographics Association Aire-la-Ville, Switzerland, Switzerland, pp. 81–90.

[TWS06] THOMASZEWSKI B., WACKER M., STRASSER W.: A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), Eurographics Association Aire-la-Ville, Switzerland, Switzerland, pp. 107–116.

[VB05] VILLARD J., BOROUCHAKI H.: Adaptive meshing for cloth animation. *Engineering with Computers 20*, 4 (2005), 333–341.

[VL03] VOLKOV V., LI L.: Real-time refinement and simplification of adaptive triangular meshes. *Visualization, 2003. VIS 2003. IEEE* (2003), 155–162.

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J.: Curved PN triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), ACM New York, NY, USA, pp. 159–166.

[ZY01] ZHANG D., YUEN M.: Cloth simulation using multilevel meshes. *Computers & Graphics 25*, 3 (2001), 383–389.