Title of the Thesis

Thesis subtitle

Master Thesis

Antoine Demont

University of Fribourg

Month and Year

Abstract

Abstract (max. 1 page)

Prof. Dr. David Bommes, Computer Graphics Group, University of Bern, Supervisor
Valentin Nigolian, Computer Graphics Group, University of Bern, Assistant

Contents

1
Introduction

2
Related work

In this chapter, we will briefly present the paper on which we based this project as well as discuss the work related to it. This contains the topic of remeshing, the introduction of a different space mapping for meshes and the two types of mesh used for object representation.

This project is based on the work of Wicke et al.[4], who present a implementation of a dynamic local remeshing algorithm and its use in physically-based simulations of elastoplastic materials.

Remeshing is a wide field but we can identify two main approaches to the problem, global or local remeshing. The first consist of remeshing the entire domain when critical conditions are met. This approach is used in the work of Bargteil et al.[1] as well as Wojtan and Turk[5]. A disadvantage of this way of tackling remeshing is that regardless of the quality of the "good" elements, the complete mesh has to be rebuilt from the ground up, inducing inefficient operations. This is especially the case where only a local area of the mesh is in bad condition whereas the rest of it still is in an good or acceptable shape. Wicke et al. [4] also mention that global remeshing accumulates large numerical errors in simulations, called *artificial diffusion*, due to the need to resample physical properties from an old to a new mesh. This can be problematic with meshes composed of materials with multiple physical properties. Those might overflow on one another, creating visual artifacts.

In the second approach, local remeshing, we try to identify the elements of bad quality from the mesh such that we then are able to replace only those during the remeshing process. This method has the benefit of minimizing the changes on the mesh while also allowing optimizations on critical parts of the mesh. Some improvements can consist for example in coarsening the mesh where it is unnecessarily detailed or inversely adding details in smaller parts of the mesh. This method also mitigates the *artificial diffusion* mentioned above, as it diminishes the need to update the physical properties since only a fraction of the mesh is actually rebuilt.

2.1 World space and material space

To ensure the bijection of all operations, we maintain a copy of the mesh, thus splitting the mesh into a world space mesh and a material space mesh (fig. 2.1). On the world space mesh will be applied all the deformations of the simulation whereas the material space will remain unchanged. Each remeshing

operation is applied to both spaces and has to be valid for each of them. This bijective state guarantees that every operation can be reverted to reset the world space mesh to its initial space. It can be used for simulating elasticity or any other temporary deformation.

Figure 2.1: TODO:Placeholder Relation between material and world space

2.2 Types of Meshes

Wicke et al.[4] is using a tetrahedral mesh for the representation of 3-dimensional objects. In this work, we will adapt their algorithm to both 2-dimensional and 3-dimensional meshes.

For the 2-dimensional representation, we will be using a triangular mesh. Here, we will render the surface of our object by assigning vertices to match its overall shape and triangulate between them to fill the surface representation. The more vertices used, the closest the representation will match the original shape. For tetrahedral meshes, not only do we render the surface but also the inside of the object. Allowing us to dig in it. A visual comparison of the two mesh types is shown in the figure 2.2.

The dimensional difference means that an object represented using a triangular mesh will only be composed of its surface, making it completely hollow, where one using a tetrahedral mesh is filled. Since a 2-dimensional mesh does not have to care about the inside of the object, it is a more lightweight and simple mean of representing it. For accurate simulations, using a 3-dimensional mesh is recommended as it can model different properties between the surface and the inside of the object.

(a) TODO:Placeholder Tetrahedral mesh

(b) TODO:Placeholder Triangular mesh

Figure 2.2: The two types of meshes used in this project

3
Mesh deformations

This chapter presents the different metrics used for evaluating the quality of a mesh and its elements, both in two and three dimensions. It also illustrate the occurrence of bad elements in a mesh as well as a first insight into how to treat with them.

3.1 Quality evaluation

Before defining what a bad element is, we need a way to measure its quality. For triangular meshes, we adapt the formula from Wicke et al.[4] to two dimension. For tetrahedral meshes, we introduce an other metric for quality: the *symmetric dirichlet energy*.

3.1.1 Triangular meshes

In [4], the quality of a tetrahedron is equal to:

$$6\sqrt{2}V\frac{\ell_{harm}}{\ell_{rms}^4} \tag{3.1}$$

with V the volume, ℓ_{harm} the *harmonic mean* of the tetrahedron's edge lengths and ℓ_{rms} the *root-mean-squared* edge length. This measure is bounded in $[0, 1]$. A quality of 1 is attributed to an equilateral tetrahedron and 0 to a degenerate one, where all the vertices are coplanar.
We find a similar approach for triangles in [2] by Freitag. The quality of a triangle is measured by:

$$4\sqrt{3}A\frac{1}{\sum_{i=1}^{3} l_i^2} \tag{3.2}$$

with, similar to eq. 3.1, A the area and l_i the ith edge of the triangle. Equivalent to eq. 3.1, a quality of 1 means an equilateral triangle and 0 for a degenerate triangle. To detect inverted triangles, we compute the signed area of the triangle such that any value of $A < 0$ indicates an inverted triangle.

Figure 3.1: TODO:Placeholder Triangle loosing quality

3.1.2 Tetrahedral meshes

For tetrahedral meshes, we chose a different approach to measuring the quality of the elements constituting the mesh. For each tetrahedron, the *symmetric dirichlet energy* is computed between it and a regular tetrahedron. In [3], the dirichlet energy is presented as a metric that evaluates the intrinsic stretching between a "source" M_0 and a "target" M surfaces in $\mathbb{R}^3$. We adapt this approach to tetrahedra by taking a tetrahedron t from the mesh and compute the dirichlet energy with a perfect regular tetrahedron.

In mathematical terms, this translates to

$$E_{dirichlet}(t) = \|\mathcal{J}_t\|^2 + \|\mathcal{J}_t^{-1}\|^2 - 6 \tag{3.3}$$

with $\mathcal{J}_t$ the *jacobian matrix* of the tetrahedron t, scaled by a coefficient of the volume, to reduce the penalty of smaller regular tetrahedra.

To note is that the reference tetrahedron can be adapted. For example, one could set the reference tetrahedron's value to the best of the mesh before any deformation is applied, thus measuring more closely the distortion between the current and initial state of the mesh.

With this method, a perfect element, meaning an element that has no stretching with respect to the reference has a quality of 0. For any other element, we invert the positive value given by the *dirichlet energy*. With this, the bigger the stretching, the worse the quality of the element will be. Any degenerate or inverted tetrahedron will have an infinitely negative quality.

Figure 3.2: TODO:Placeholder Tetrahedron loosing quality

3.2 Bad elements

Now that we can identify bad elements in a mesh, we will discuss when and how those can appear. In figure 3.3, we see that when bending the mesh, some of the geometry starts to overlap. Since we move vertices of the mesh without constraints, the vertices of some elements will move pass their neighbours. Thus creating first degenerate and then inverted elements.

By performing remeshing operations, we aim to keep the shape of the object but change the geometry of the bad elements to stay within acceptable quality values.

An other option would be to limit the vertices' movements to avoid creating low quality elements but this approach would mean that some deformations could not be applied to the object. Since some would inevitably create low quality elements on the mesh.

Figure 3.3: TODO:Placeholder Object bending with inverted elements

4 Remeshing

The chapter 4 covers the remeshing algorithm as well as its implementation in two and three dimensions.

4.1 General remeshing algorithm

The overall operation of the algorithm, presented in alg.1, goes as such: We first have to evaluate the quality of the mesh. This is done by using the formulas presented in chapter 3 to compute the quality of each element present in the mesh. If one's quality is below an empirically fixed threshold q_{min}, then we add it to a list of elements to perform the remeshing on. Once we have found all the critical elements, an iterative improvement is performed on each of them. This operation is described in alg. 2.

Algorithm 1 Loop: The overall remeshing algorithm

Require: M a mesh to improve
  $B \leftarrow \{\}$
  for all $t \in M$ **do** ▷ Find bad elements
    $q \leftarrow ComputeQuality(t)$
    if $q < q_{min}$ **then**
      $B \leftarrow B \cup t$
    end if
  end for
  for all $b \in B$ **do** ▷ Improve mesh
    $ImproveElement(b)$
  end for

The remeshing is composed of three sub-parts, in order the *Topological*, *Contraction*, *Insertion* and *Smoothing* passes. We also see a list called A, sorted based on its elements' quality. It will reduce and expand at each pass. A contains at first only the element of bad quality we want to get rid of but will change in size after each mesh operation. Since by changing the mesh we also change the neighbours to the starting element, an improvement can be required on those too. To note is A may not only contain

elements of quality below q_{min}, Wicke et al.[4] mention that their experiences showed some bad elements can only be improved by also altering their neighbours.

Algorithm 2 ImproveElement: Mesh improvement algorithm

Require: t an element of bad quality, N the number of remeshing iterations
function IMPROVEELEMENT(t)
$A \leftarrow \{t\}$
for $i \leftarrow 0$ to N **do**
do
$A \leftarrow TopologicalPass(A)$
if $Quality(A) \geq q_{min}$ **then** ▷ $Quality(A)$ returns the worst element of A
return
end if
while $TopologicalPass(A)$ changes the mesh
$A \leftarrow ContractionPass(A)$
if $Quality(A) \geq q_{min}$ **then**
return
end if
$A \leftarrow InsertionPass(A)$
if $Quality(A) \geq q_{min}$ **then**
return
end if
$A \leftarrow SmoothingPass(A)$
if $Quality(A) \geq q_{min}$ **then**
return
end if
end for
end function

function CONTRACTIONPASS(A)
$E \leftarrow$ set of all edges of elements in A
for all $e \in E$ **do**
if e still exists **then**
attempt to contract e, smooth resulting vertex
end if
end for
return the surviving elements of A and the mesh elements modified by edge contraction
end function

To improve an element t of the mesh, alg. 2 uses an iterative process where a list of elements goes through a succession of passes to locally improve quality. To note that [4] uses a value of $N = 10$ but it was adapted in this implementation due to performance. Since their implementation differs based on the dimensionality of the mesh, the details of the *Topological*, *Insertion*, *Smoothing* passes will respectively be discussed in sections 4.3 and 4.4. As for the *Contraction pass*, the idea is to try to contract each element's edges to try to get rid of those that are close to being degenerated. After each edge contraction, we smooth the resulting vertex and evaluate the quality of the remaining elements near it. We then can update the mesh or revert the operation based on the new local quality. This process is illustrated in figure 4.1.

Figure 4.1: TODO:Placeholder Flow diagram for edge contraction

4.2 Control mesh

As presented in chapter 2, we represent the object in two distinct spaces called world and material space. To its representation in world space are applied all the deformation whereas in material space only the topological changes are applied. This method ensures bijection on all mesh operations.

This means that, when we perform a remeshing step on the world mesh, it also needs to be valid on the control mesh in the material space. As illustrated in fig.4.2, after each operation is performed we check whether it creates a bad element in the control mesh. In this case we reset the material space mesh to its previous stable state or update the control mesh if all elements are of sufficient quality.

With this, each modification made on the mesh has an opposite operation able to revert it to its original state.

Figure 4.2: TODO:Placeholder Flow diagram for control mesh validation

4.3 Triangular meshes

This section describes the implementation of the topological, insertion and smoothing passes with triangle meshes.

4.3.1 Topological pass

In three-dimensional meshes, Wicke et al.[4] use this pass to remove edges and faces of bad quality. We adapted it to two-dimensional meshes while keeping this idea of low quality element removal. All without being redundant with the contraction pass presented in alg. 2. To do so we used the edge flip.

Shown in fig. 4.3, we see that by flipping an edge both of the bad quality faces disappear to leave only elements of an acceptable quality. Now that we have a mesh operation similar to the one in [4], the algorithm (alg. 3) for the *topological pass* goes as such:

We take the edges of all elements we want to improve quality. For each of them, a flip is performed and we check whether the quality of the neighbouring triangle improves. To note is that boundary edges cannot be flipped as it would modify the shape of the object and not only its topology. For all triangles whose quality improved, we add them to a set of new triangles, even if their quality is above the quality threshold. This set can then be processed in another pass of alg. 2.

Figure 4.3: TODO:Placeholder Edge flip improving triangle quality

Algorithm 3 Topological pass on triangular meshes

Require: A a set of triangles of the mesh
  function TOPOLOGICALPASS(A)
    $changed \leftarrow false$
    $E \leftarrow$ set of all edges of triangles in A
    $A_{new} \leftarrow \{\}$
    for all $e \in E$ **do**
      if e still exists **then**
        attempt to flip e ▷ t_1, t_2 created by flipping edge e
        $A_{new} \leftarrow A_{new} \cup t_1, t_2$
        $changed \leftarrow true$
        if $Quality(t_1, t_2)$ did not improve **then** ▷ Ensure mesh improvement
          revert operation
        end if
      end if
    end for
    return the surviving elements of $A \cup A_{new}$ and $changed$
  end function

4.3.2 Insertion pass

In this section, I'll present the cavity digging and filling in 2D.

4.3.3 Smoothing pass

In this section I'll cover the remeshing pass, showing a point going to the average of its neighbours.

Figure 4.4: TODO:Placeholder 2D Cavity building with successive circumcircles

4.4 Tetrahedral meshes

This section describes the implementation of the topological, insertion and smoothing passes with tetrahedral meshes.

4.4.1 Topological pass

In this section, first I will show the different types of flips and mesh operations. I'll then describe the topological pass' algorithm, with the option between face and edge removal.

4.4.2 Insertion pass

In this section, I'll explain the star system to incrementally add tetrahedra to a set of stars, called a galaxy, with the goal of digging cavities inside of the mesh. Then, I'll cover the algorithm for the complete pass with cavity digging, filling them up and the final topological passes on the added elements. I also need to illustrate the center of chebyshev.

4.4.3 Smoothing pass

In this section, I'll describe the algorithm for the smoothing pass. Using the above mentioned center of chebyshev. Not forgetting to mention that it is the only step that does not guarantees global improvement (only local) but may allow future operations to occur.

5 Results

This chapter describes the different experiments plus discusses their quality.

5.1 Triangular meshes

In this section, I'll describe the two experiments made in 2D and show the search for the optimal value of minimal quality. Mentioning that 2D was the stepping stone into the project and not many experiments were run.

5.1.1 Stretch

Figure 5.1: TODO:Placeholder Stretch experiment

5.1.2 Compress

Figure 5.2: TODO:Placeholder Compress experiment

5.1.3 Finding q_{min}

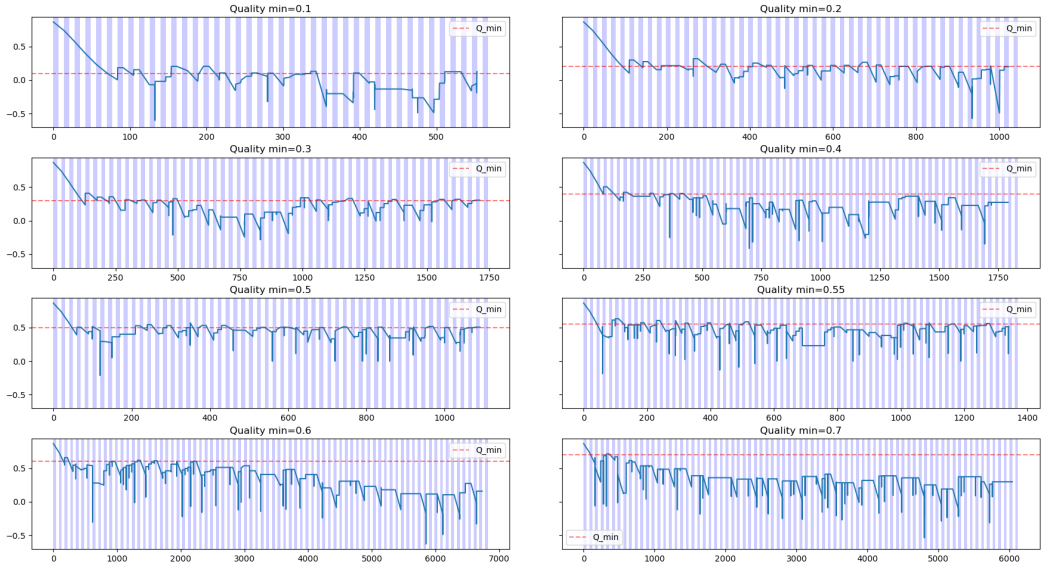


Figure 5.3: Evolution of the quality for different tresholds

5.2 Tetrahedral meshes

In this section, I'll go through the different experiments done in 3D.

5.2.1 Spin

Figure 5.4: TODO:Placeholder Spin experiment

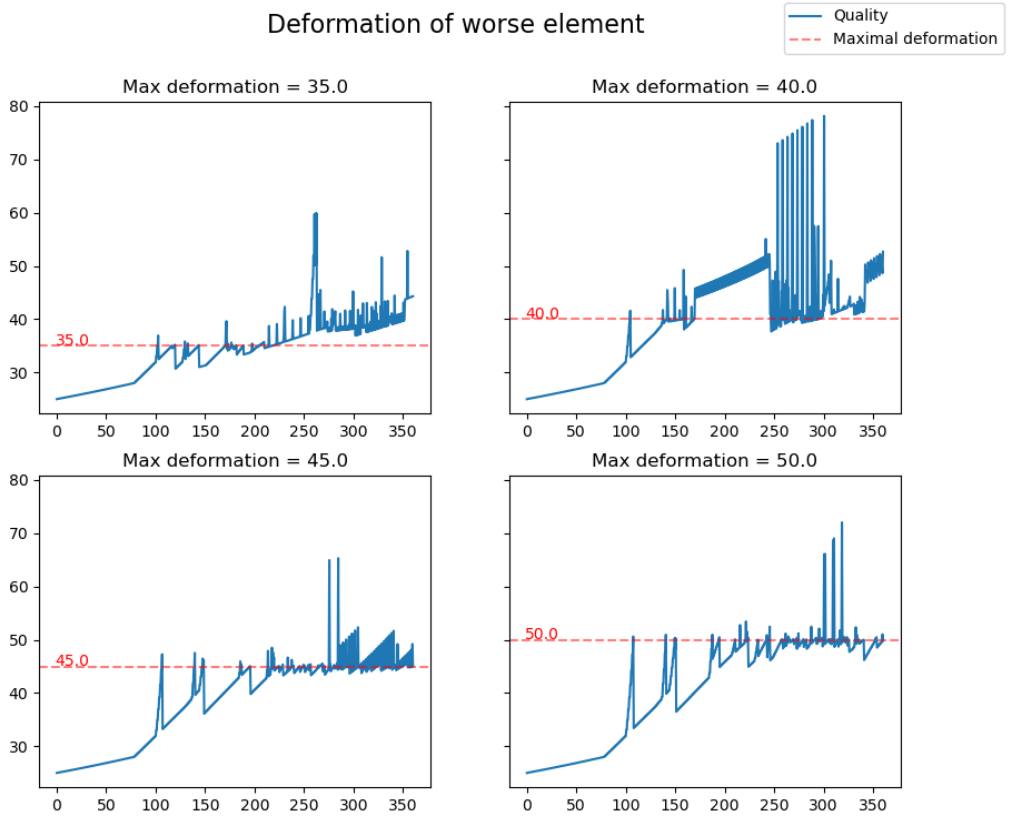
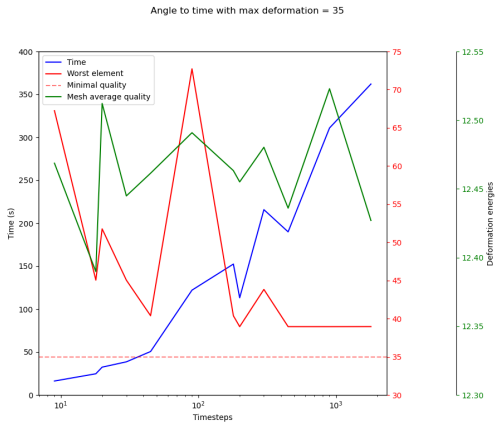
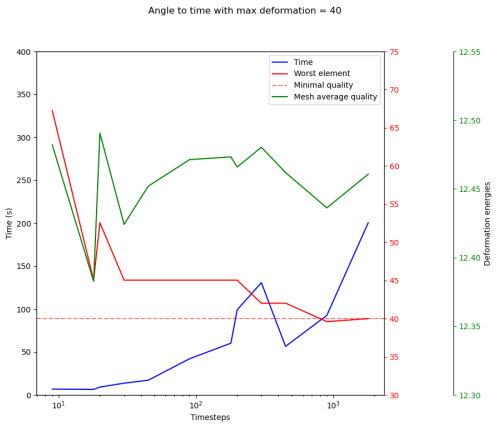


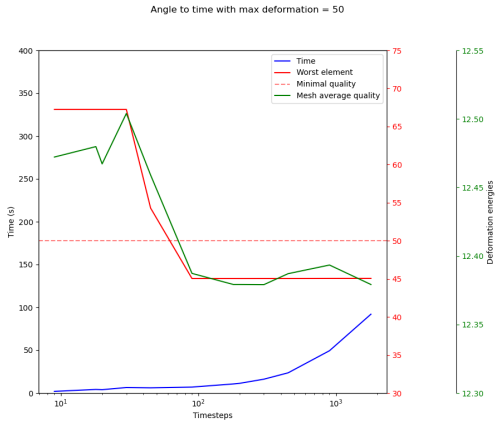
Figure 5.5: Finding the optimal quality



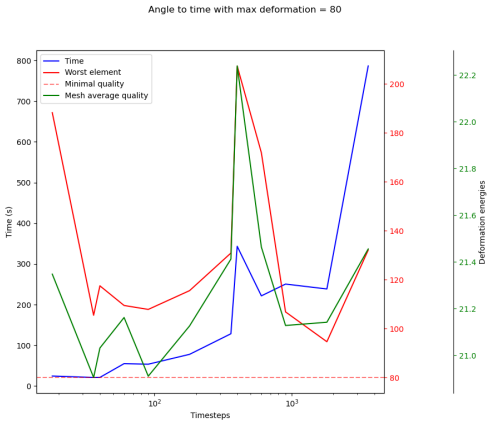
(a) Half turn with maximal deformation of 35



(b) Half turn with maximal deformation of 40



(c) Half turn with maximal deformation of 50



(d) Full turn with maximal deformation of 80

Figure 5.6: Finding the optimal timestep angle

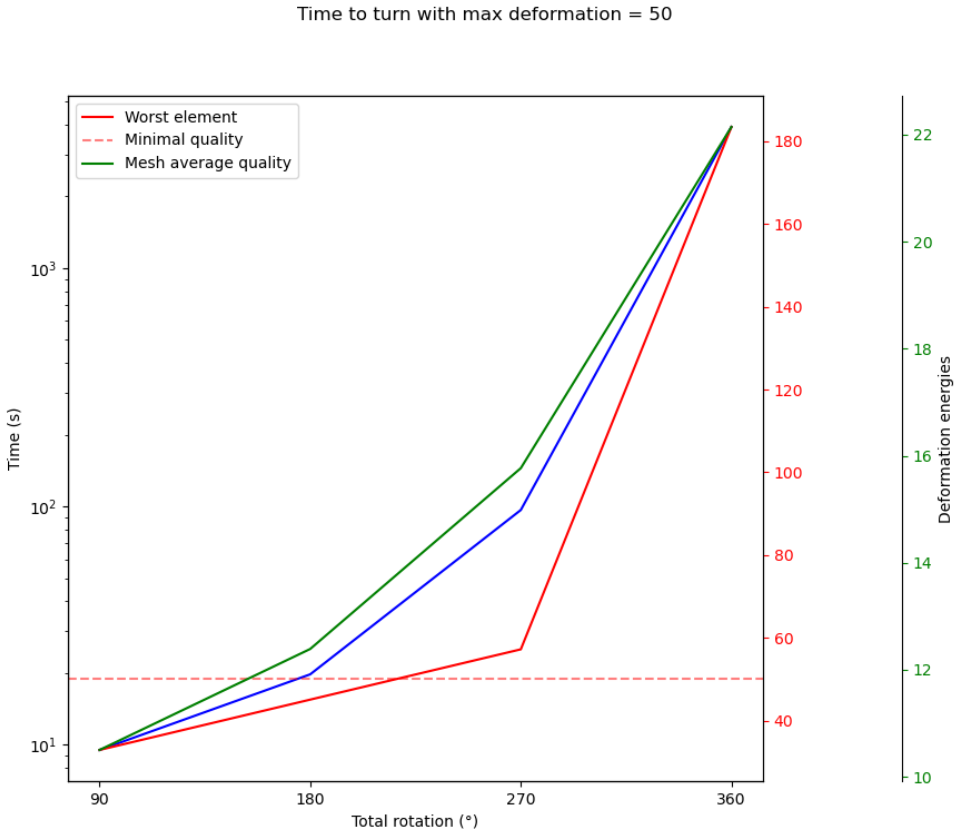


Figure 5.7: Evolution of performance with bigger deformations

5.2.2 Stretch

Figure 5.8: TODO:Placeholder Stretch experiment

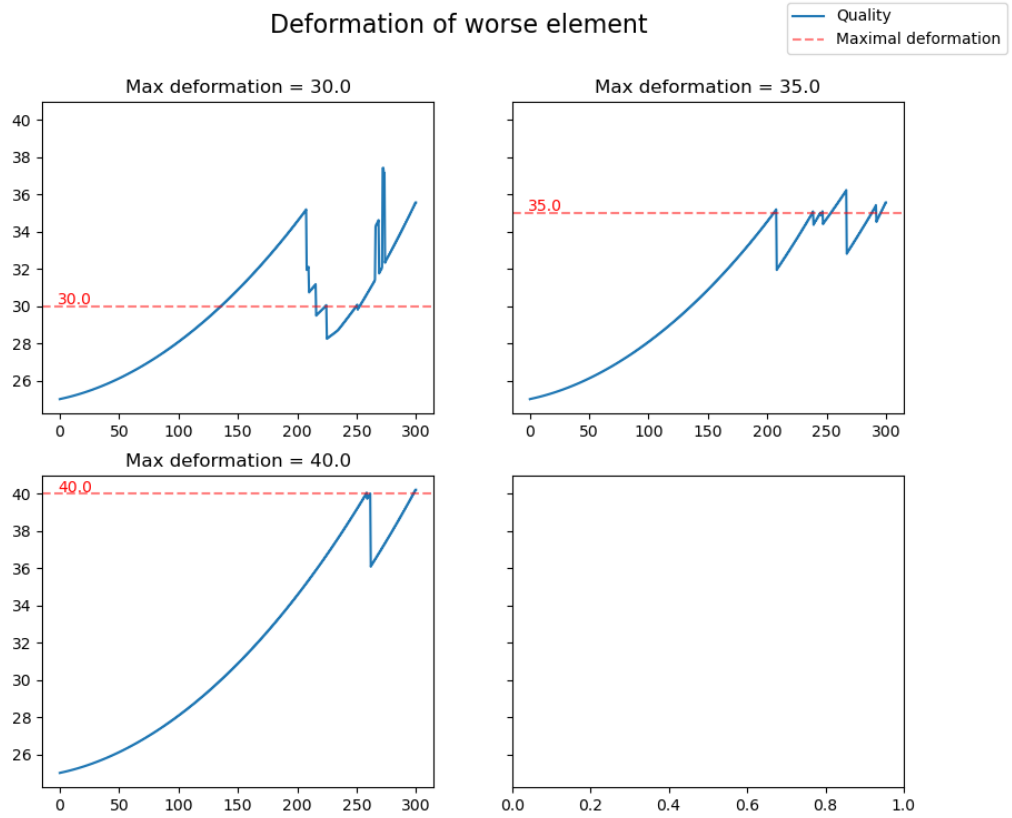


Figure 5.9: Finding the optimal quality

Figure 5.10: TODO:Placeholder Finding the optimal timestep

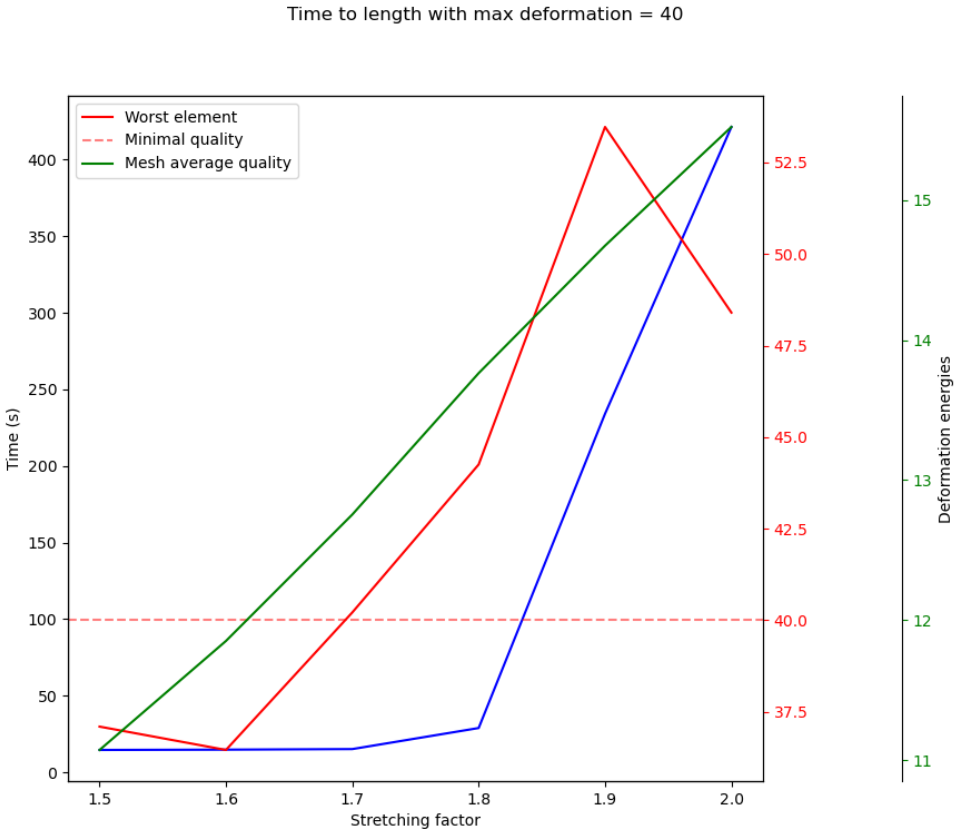


Figure 5.11: Evolution of performance with bigger deformations

5.2.3 Impact of the control mesh

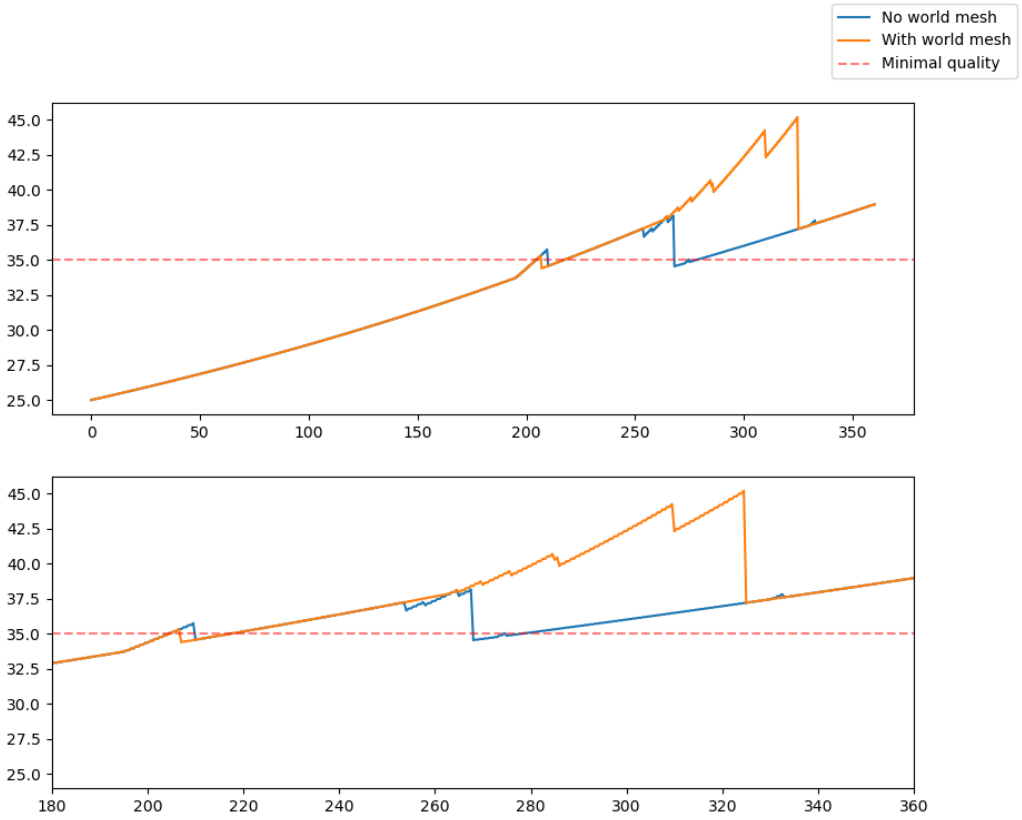


Figure 5.12: Evolution of the quality with and without using material space

6 Conclusion

C fini

Bibliography

[1] Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. A finite element method for animating large viscoplastic flow. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 16–es, New York, NY, USA, 2007. Association for Computing Machinery.

[2] Lori Freitag, Mark Jones, and Paul Plassmann. A parallel algorithm for mesh smoothing. *SIAM Journal on Scientific Computing*, 20(6):2023–2040, 1999.

[3] Justin Solomon, Leonidas Guibas, and Adrian Butscher. Dirichlet energy for analysis and synthesis of soft maps. *Computer Graphics Forum*, 32(5):197–206, 2013.

[4] Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph.*, 29(4), jul 2010.

[5] Chris Wojtan and Greg Turk. Fast viscoelastic behavior with thin features. *ACM Trans. Graph.*, 27(3):1–8, aug 2008.