MULTI-RESOLUTION REPRESENTATION OF B-SPLINE SURFACES FOR INTERACTIVE EDITING:

CAGD II – COURSE PROJECT

Lavanya Sita Tekumalla

U0320994

1. MOTIVATION

Multi level or multi resolution representation of surfaces has several advantages such as compression, hierarchical editing, progressive transmission and so on. This project focuses on a multi-level representation of surfaces through refinement for editing of b-spline surfaces. The hierarchical representation is implemented using b-spline surface refinement (uniform bi-cubic subdivision surfaces).

The advantage of having a multi level representation for hierarchical editing is that we can edit the overall geometry by modifying the coarser level control points while we can add finer detail or features by moving the finer level control points. However we need a mechanism to update the other corresponding levels consistently when we update one level of the surface.

2. PROBLEM DEFINITION

The two operations that a user can perform at any time are-

1. Editing the coarser level
2. Editing the finer level

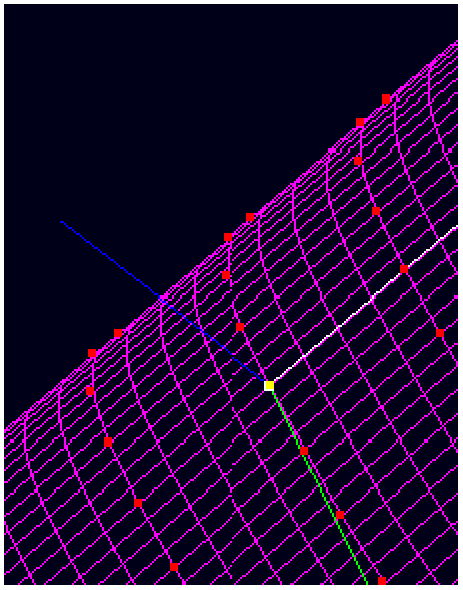
When a fine level is modified, we cannot find an equivalent surface with the coarser level basis-functions. So we need to somehow keep track of the modifications in the finer level separately. And when the coarser geometry is updated, make the appropriate changes to the finer geometry.

Another idea that has been tried is, when we update the finer level, trying and updating the coarser geometry and get the closest possible approximation to the finer geometry in the least square sense (since we cannot get an exact representation using the basis functions of the coarser level).

Also we need a means to specify the motion of control points in 3D. This has been done by computing the partial derivatives and the normal of the surface at the node value corresponding to the control point.

3.1 SPECIFYING CONTROL POINT POSITION BY DEFINING A LOCAL COORDINATE SYSTEM

Specifying motion in 3D is a difficult problem. So instead, when a user selects a control point and wants to move it, an orthonormal basis (u_1,v_1,n_1) is displayed at that point with normal n_1 pointing to the normal to the surface at that point, and u_1 and v_1 in the tangent plane at that point. This makes it intuitive and easy to specify motion.



Computing the Local basis system:

Computing $\partial s(u,v)/\partial u$: For every column j, the control points of the surface corresponding to the partial derivative with respect to u are computed as

$$\frac{(P_{i+1,j} - P_{i,j})u\deg}{(\text{uk}[\text{i}+u\text{deg}+1]-\text{uk}[\text{i}+1])}$$

Computing $\partial s(u,v)/\partial v$: For every row i, the control points of the surface corresponding to the partial derivative with respect to u are computed as $\frac{(P_{i,j+1} - P_{i,j})v\deg}{(\text{vk}[\text{j}+v\text{deg}+1]-\text{vk}[\text{j}+1])}$

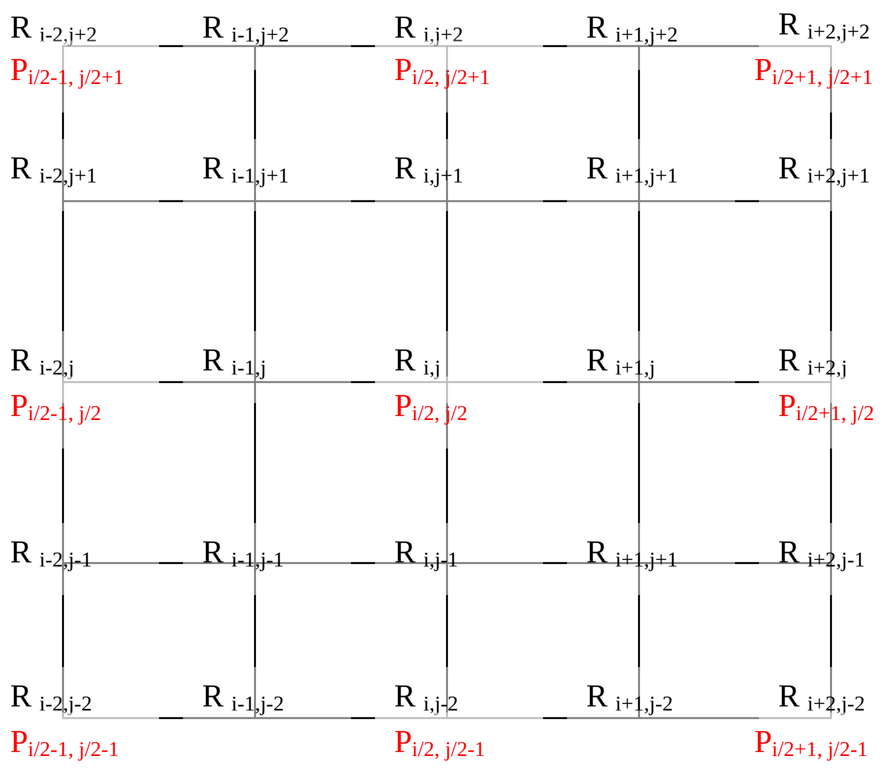
where *u*deg is the degree in the u direction, *v*deg the degree in the v direction, uk the u knot vector and vk the v knot vector.

The knot vector of the new spline surface (partial derivative) is similar to the original knot vector. It is uniform floating with two fewer elements compared to the original.

When we want to move a particular control point, the node value in the parameter space corresponding to that control point is taken – say this is (u,v)
The normal is computed as n= $\partial s(u,v)/\partial u$ x $\partial s(u,v)/\partial v$

The local basis chosen is $\partial s(u,v)/\partial u$, $\partial s(u,v)/\partial u$ x n and n. The amount of motion in each of these directions is specified by the user, using sliders in the GUI.

3.2 UNIFORM BI_CUBIC SUBDIVISION (CATMULL_CLARK SUBDIVISION)



This method proceeds by inserting a new knot between adjacent knots in the coarser surface and finding the corresponding new control points in terms of the old control points.

<u>The Subdivision Template</u>

For every control point $R_{i,j}$ on the refined mesh, the Catmull-Clark subdivision template can be expressed as follows

If i is odd and j is odd

$$R_{i,j} = P_{(i/2 + ½,\ j/2 + ½)} + P_{(i/2 + ½,\ j/2 - ½)} + P_{(i/2 - ½,\ j/2 + ½)} + P_{(i/2 - ½,\ j/2 - ½)}$$

If i is odd and j is even

$$R_{i,j} = P_{(i/2 + ½,\ j/2)} * 3/8 + P_{(i/2 - ½,\ j/2)} *3/8$$
$$+ P_{(i/2 - ½,\ j/2 + 1)} *1/16 + P_{(i/2 + ½,\ j/2 + 1)} * 1/16$$
$$+ P_{(i/2 - ½,\ j/2 - 1)} *1/16 + P_{(i/2 + ½,\ j/2 - 1)} *1/16$$

If i is even and j is odd

$R_{i,j} = P_{(i/2\ ,\ j/2 - 1/2)} * 3/8 + P_{(i/2\ ,\ j/2 + 1/2)} * 3/8$

$+ P_{(i/2 - 1,\ j/2 + 1/2)} * 1/16 + P_{(i/2 + ½,\ j/2 + 1/2)} * 1/16$

$+ P_{(i/2 - 1,\ j/2 - 1/2)} * 1/16 + P_{(i/2 + 1,\ j/2 - 1/2)} * 1/16$

If i is even and j is even

$R_{i,j} = P_{(i/2\ ,\ j/2\ )} * 9/16$

$+ P_{(i/2\ ,\ j/2 + 1)} * 3/32 + P_{(i/2\ ,\ j/2 - 1)} * 3/32$

$+ P_{(i/2 + 1,\ j/2\ )} * 3/32 + P_{(i/2 -1,\ j/2\ )} * 3/32$

$+ P_{(i/2 - 1,\ j/2 -1)} * 1/64 + P_{(i/2 - 1,\ j/2 +1)} * 1/64$

$+ P_{(i/2 + 1,\ j/2 + 1)} * 1/64 + P_{(i/2 + 1,\ j/2 - 1)} * 1/64$

3.3 FINDING THE CLOSEST ESTIMATE OF THE CONTROL POINTS OF THE BASE MESH

Let M be the number of rows in the control mesh and N the number of columns

The above template can be expressed as follows.

For every control point $R_{a,b}$ of the refined mesh,

$$R_{a,b} = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} P(i,j)w(i,j)$$

where w(i,j) is defined by the template mentioned above if $R_{a,b}$ contributes towards $P(i,j)$,and 0 otherwise. In other words, for all R_i where R_i is a control point belonging to the refined mesh,

$$R_i = \sum_{j=0}^{MN-1} P_j * w_j$$

This is a linear system and can be expressed as

$$W\,P = R$$

Where P is a (MN x 1) matrix, R is a ((2M-3 * 2N-3) x 1) matrix and W is a ((2M-3 * 2N-3) x MN) matrix

Given a refined mesh, the *R* matrix is known, and the *W* matrix is known from the subdivision template. We can try to solve this system of equations and try to obtain a base mesh corresponding to this refined mesh. However, the above linear system is over determined. Hence Singular Value Decomposition has been used to solve this linear system in order to get the closest base mesh in the least squares sense.

4 IMPLEMENTATION

At any point of time, there are two surface objects that the user can access, the base surface B and the refined surface S. Also In the program, every refined surface object has associated with it a base surface object (Sb). This surface Sb is a data member of the refined surface.

Let **u** be the unit vector in the direction of partial derivative of the surface with respect to u direction at that point. Let **v** be the unit vector in the direction of partial of the surface with respect to v direction at that point and let **n** be the unit vector in the direction of cross product of u and v (perpendicular to the tangent plane)

<u>Modifying Base Surface B --> B'</u>

The control points of the surface B are just set to the new values corresponding to B' as specified interactively by the user.

However, this change needs to be reflected in the refined mesh S too. Since the refined surface S might have been updated in the meantime, the amount by which it has been updated is obtained as follows:

Its base surface Sb is invoked and subdivided the adequate number of times to obtained the refined mesh Ss (This would have been our refined mesh if S has not been updated at any point of time). Each control point of Ss and S are compared, and their difference is split into three components along the directions **u**, **v** and **n** at that point. These differences are stored in a matrix D

Now the new base mesh B' is copied into Sb and S is obtained by refining Sb the adequate number of times. Now the control points of S are modified according to the matrix D. In other words, each control point of S is displaced along its **u**, **v**, and **n** directions according to the values stored in D. We thus get our new refined surface S.

When Refined Surface is Updated S -> S'

The control points of the surface S are just set to the new values corresponding to S' as specified interactively by the user.

There is a checkbox to specify the option of attempting to update the base mesh adequately. If this box is checked, the base mesh is changed so that it is as close to the refined mesh as possible. Now B and Sb are updated to this new base mesh.

5 SOLVING AN OVER DETERMINED SET OF EQUATIONS

Suppose we have a set of equations AX=B .

The number of unknowns (dimension of X) is n and the number of equations is m. In our case, m>n.

||AX-B|| is called the residue .

Since the system is over determined, the matrix A is rectangular and is not invertible. Under these circumstances, commonly, we try to find a solution that minimizes the least square norm $|Ax-B|^2$. When we equate the partial derivative of this expression with respect to x to 0, we end up with the expression

$A^tA\,x - A^tB = 0$

Or $A^tA\,x = A^tB$

These are called the normal equations.

And $x = (A^t A)^{-1} A^t B$

The Expression $A^+ = [(A^t A)^{-1} A^t]$ is called the pseudo inverse of the Matrix A. A^+ is also called the Morse Penrose inverse – since A^+ satisfies $A A^+ A = A$

Finding the Pseudo inverse of a Matrix

Finding the inverse of a matrix by the usual method (Adj(A)/Det(A)) is an extremely tedious process and is hardly ever used.

The following are some of the common techniques employed in order to solve our system $Ax=B$

Cholesky Factorization

Every positive definite matrix A can be factored uniquely into the product $A=L^t L$ where L is a lower triangular matrix. This is the Cholesky factorization.

- Split $A^t A$ into $L^t L$ where L is a Lower triangular matrix.
- Let $W= Lx$
- Solve the lower triangular system $L^t W = A^t B$
- Now solve the upper triangular system $L x = W$
- The time complexity of this algorithm is of the order $O(mn^2 + n^3/3)$

QR Decomposition:

QR factorization is a more modern way of solving the least squares problem. Given a matrix A, it can be decomposed into an upper triangular matrix R and an orthogonal Matrix Q ($QQ^t=I$)

- Decompose A as $A= Q R$
- $Q R x = B$
- $R x = (Q^t B)$

This is an upper triangular system and can be solved by back substitution. This method gives us a technique for finding the least squares approximation to the solution Complexity ($2mn^2 – 2/3\ n^3$)

Singular Value Decomposition

Every m x n matrix A with m>n can be decomposed into

$A = U D V^t$

Where D is a diagonal matrix and U and V are orthogonal matrices.

Our system of equations is $A\ x = B$

- Obtain the singular value decomposition of A: $A = U\ D\ V^t$
- $U\ D\ V^t x = B$
- $x = (U\ D\ V^t)^{-1} B$
- $x = (V\ D^{-1}\ U^t)\ B$

$A^+ = (V\ D^{-1}\ U^t)$ [the pseudo inverse of A]

Complexity : $2\ mn^2 + 11\ n^3$

Comparison of methods:

Condition number is the ratio of the largest singular value to the smallest singular value and when the condition number is too large, the matrix is ill-conditioned.

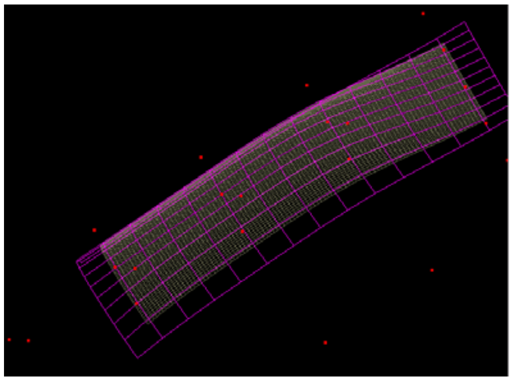
Solving by Cholesky factorization is more efficient than SVD. However, for ill-conditioned matrices, the condition number of A^tA is the square of condition number of A (The eigen decomposition of A^tA is VD^2V^t if singular value decomposition of A is UDV^t) . Hence solving the normal equations directly might have numerical instability issues. Hence SVD or QR factorization methods are generally preferred.

When the rank of A is equal to n, the least squares problem has a unique solution. However when $\mathrm{rank}(A) < \min(m, n)$ -- in other words, *A* is rank-deficient - we seek the minimum norm least squares solution x which minimizes both $||x||^2$ and $||Ax-B||^2$. And SVD provides such a solution. For matrices that are not rank deficient, QR is the preferred method as it is cheaper than SVD when m and n are almost equal.

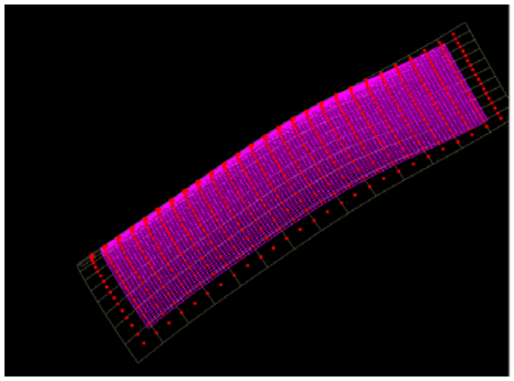
However, When $m>>n$, the two methods have almost the same time complexity. And SVD is the method used in this project

6. RESULTS

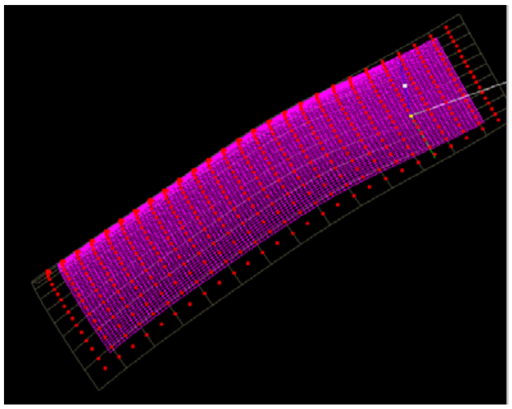
1. Finding the refined mesh when the base mesh is updated



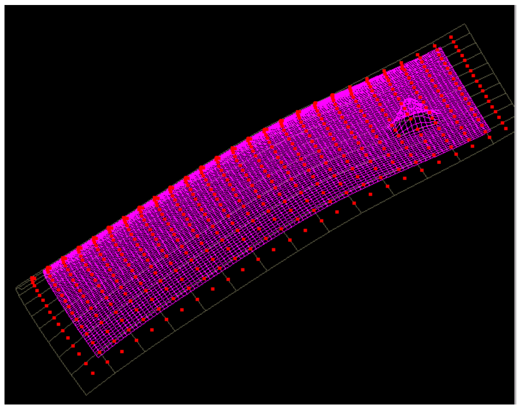
The original base mesh



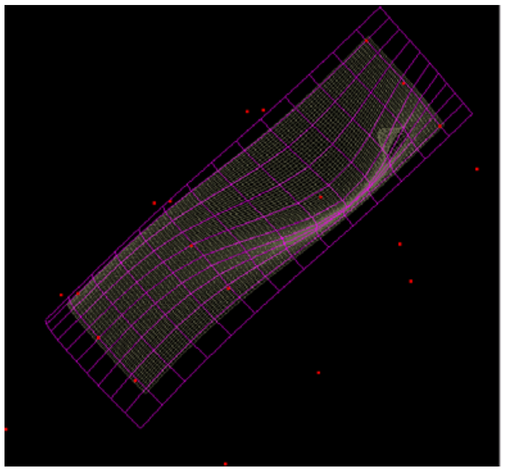
The original refined mesh



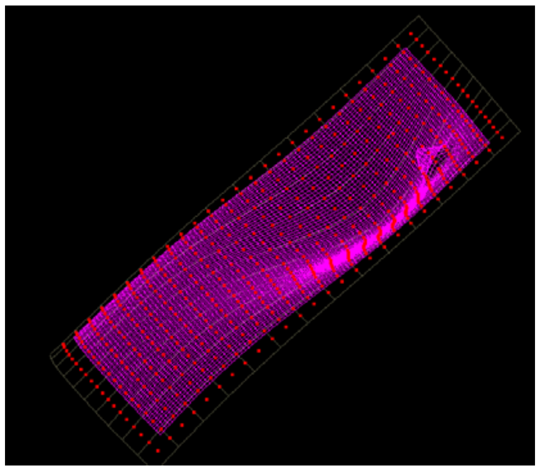
A control point in the refined mesh being moved



New refined mesh after a control point in the refined mesh has been moved

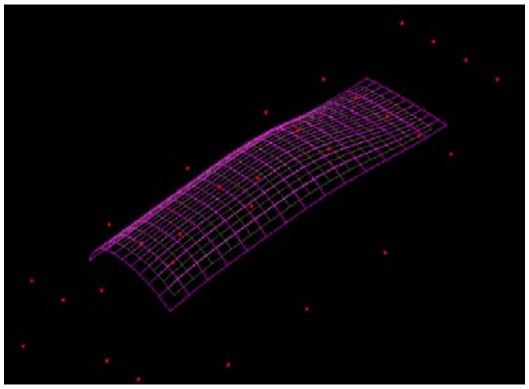


New base mesh after a control point in the base mesh has been moved

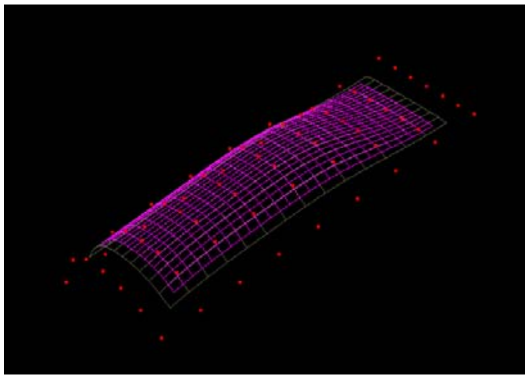


New refined mesh reflecting the change made to the base mesh and retaining its detail previously added.

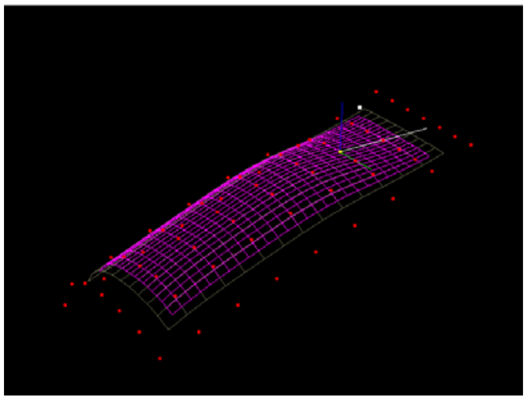
2. **Finding the closest base mesh given the refined mesh**



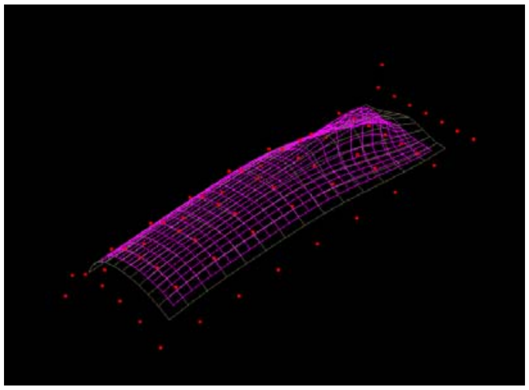
Original Base mesh



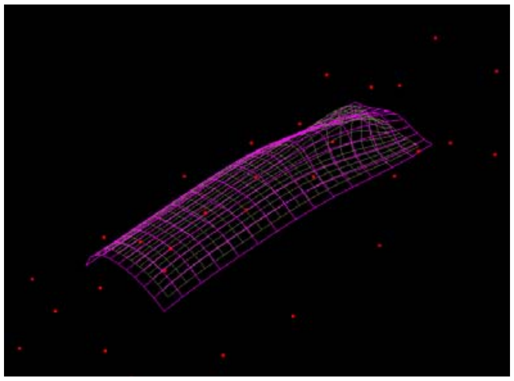
Original Refined mesh



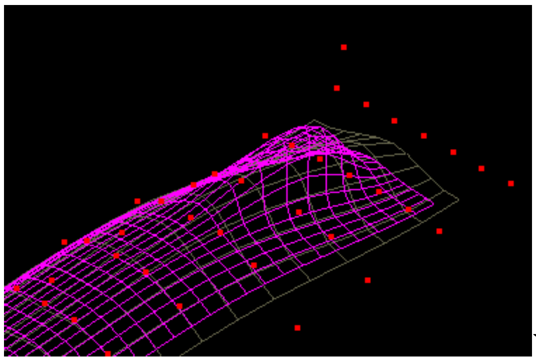
Refined mesh being changed (A control point is being moved)



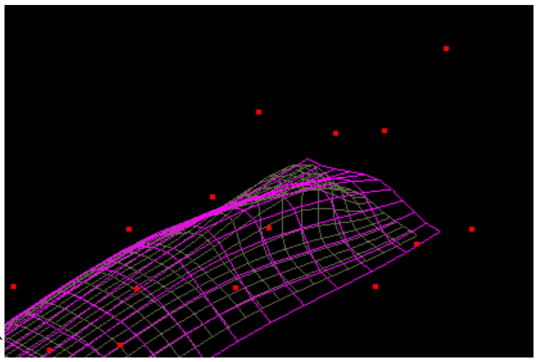
New refined mesh



New base mesh- updated going by the idea previously described

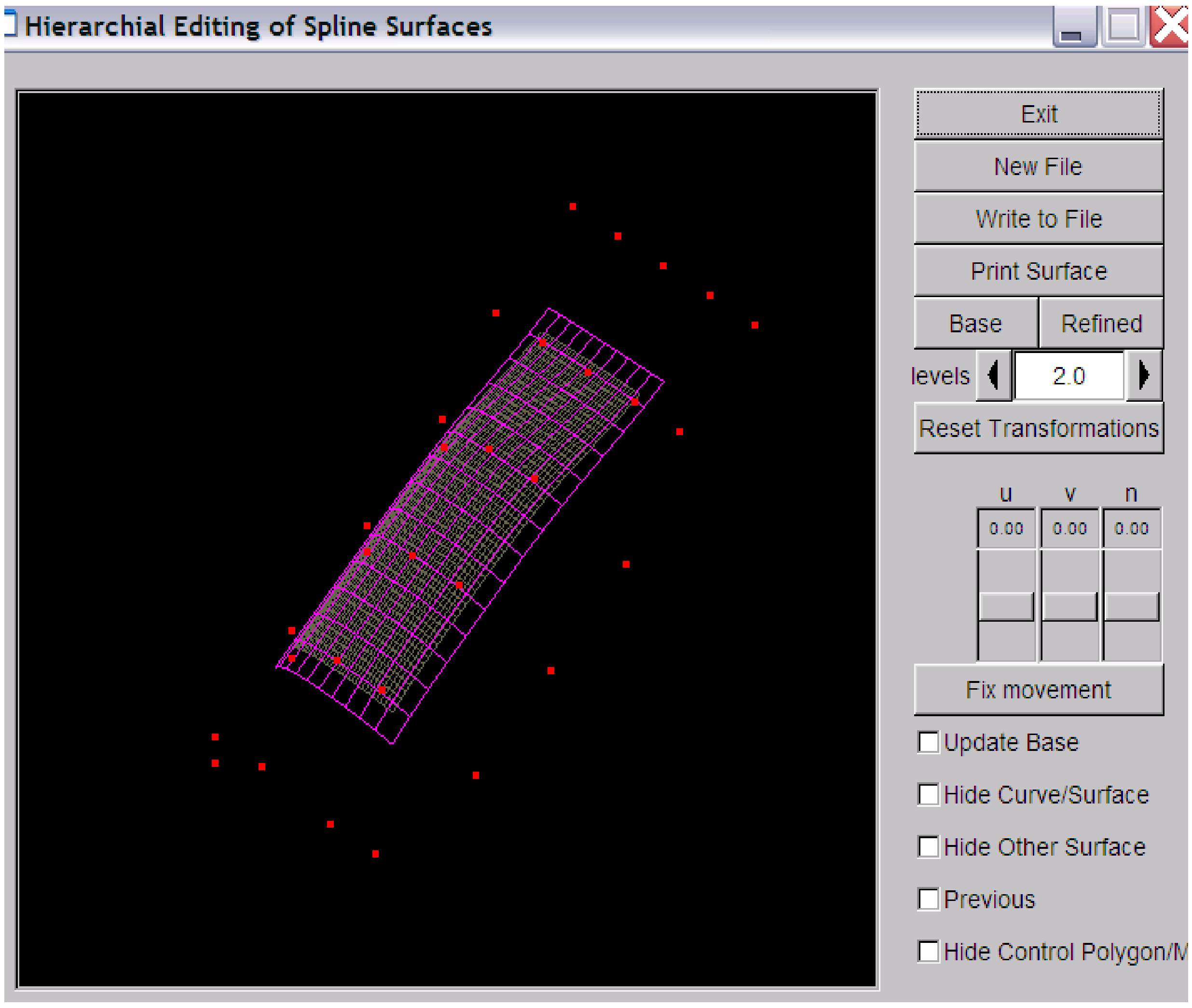


A closer view of the new refined mesh



A closer view of the new base mesh- updated going by the idea previously described

7. THE GUI



Exit: Exit the application

New File: Read a new base mesh and get the appropriate refined mesh

Write To File: Write the current mesh base/refined to file.

Print Surface: Print the control mesh and knot vector of the current object to console

Base: Show the base mesh

Refined: Show the refined mesh

Levels: Specify how many levels the refined mesh must be subdivided from the base mesh.

Reset Transformations: Reset all the transformations done to the object so far (rotation, translation and scaling)

u, v and **n**: Specify how much to move a control point along its u, u x n and n directions

Fix movement: Fix the movement specified by the u, v and n sliders and update the surfaces

Update Base: Update the base mesh when the refined mesh is modified

Hide Curve/Surface: Do not show the surface when checked

Hide other surface: Do not show the refined mesh when the current object is the base mesh, and vice versa

Previous: When we modify the current object, when this box is checked, the object prior to the modification is shown in a different color.

Hide Control Polygon: Hide the control polygon or the control mesh of the surface.

The object under consideration can be rotated by moving the left mouse button, zoomed using the right mouse button and translated using the center mouse button. Any control point can be selected by clicking on it and it can be moved along **u, v** and **n** as specified using the sliders provided

References:

Elaine Cohen, Richard F Riesenfeld and Greshon Elber, "Geometric Modelling with Splines- An Introduction"

Numerical Recipes - http://www.library.cornell.edu/nr/cbookcpdf.html