A Moving Least Squares method for implant model deformation in Computer Aided Orthopedic Surgery for fractures of lower extremities

Esmitt Ramírez[1] and Ernesto Coto[1]

[1]Centro de Computación Gráfica, Universidad Central de Venezuela, Caracas

ABSTRACT

Preoperative planning is an essential step before performing any surgical procedure. Computer Aided Orthopedic Surgery (CAOS) systems are extensively used for the planning of surgeries for fractures of lower extremities. These systems are input an X-Ray image and the planning can be digitally overlaid onto the image. The planning includes reassembling the fractured bone and possibly adding implants to reduce the fracture. In many cases, the implant does not fit perfectly in the patient's anatomy and it must be bended to adjust the implant to the bone. This paper presents a new method for the deformation of implants in CAOS systems, based on the Moving Least Squares (MLS) method for 2D images. Several improvements over the original MLS method are introduced to achieve visual results similar to the real procedure and make the deformation process easier and simpler for the surgeon. The improvements are explained in detail and all parameter values are provided. Over 100 clinical surgeries have been already planned successfully using a CAOS system that employs the proposed technique.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; I.3.8 [Computer Graphics]: Applications—

1. Introduction

Preoperative planning is a vital step that all surgeons should follow prior to performing any surgical procedure. The main purpose of this planning is determining the final result of the surgery and setup the surgical technique to apply during surgery. As explained in [SSK07], preoperative plans for fractures can be done manually by tracing on paper the patient's fractured bones using previously captured standard radiography images. When implants are required, they are traced too using clear plastic templates of the implants.

Although this manual procedure has proven useful, it is very time consuming and error prone. In past years, a variety of software systems used on orthopedic surgeries have been developed, where the preoperative planning can be done digitally, see [YSTS*09] for an example. These kinds of systems are called CAOS (*Computer Aided Orthopedic Surgery*). In a CAOS system the surgeon can load X-Ray images and digitally reassemble the pieces of the fractured bone. If an implant is required, the surgeon can select one from an implant library and overlay it over the X-Ray image. The surgeon can also add annotations and measurements which could be of help during surgery. The process is completely digital and so the preoperative surgery plan can simply be printed out and taken into the surgery room.

In some cases the surgeon needs to bend the implant to make it fit in the correct anatomical position. This bending is done in the surgery room while the patient is laying on the surgery table, and can be done several times until the implant fits correctly. This step could also be planned digitally so as to avoid this repetitive manual bending, therefore reducing surgery time.

Previously, we presented a CAOS system which included an implant deformation stage, see [RC10]. This stage consisted on a 4-step pipeline. The first step is loading the 3D model of the implant from an implant library. All models in the library are in STL (*stereolithography*) format. The second step projects the model using parallel projection onto the visualization plane. In this stage it is possible to project the implant from six different viewpoints (top, bottom, right, left, front and back). The third step is rendering this projection and overlaying it on the patient's X-Ray. In the final step a set of point handlers are located along the major axis of the implant image. Using these handlers the user can bend the implant.

In [RC10] we used the warping technique presented in [BJ03], but this did not produce clinically acceptable visual results, since this technique deforms the complete image while in practice surgeons require only local deformations. In this paper, we present a new method for implant deformation for our CAOS system, which is specifically focused on fractures of the lower extremity. This method is a new variant of the Moving Least Squares (MLS) approach for 2D image deformation [SMW06]. Our proposed method includes the automatic placing and distribution of deformation handlers, an improved strategy to manipulate the handlers and a new weight function for the MLS approach, which improves deformation results. The presented method does not generate foldbacks and the deformation is very similar to the real bending in the surgery room.

This paper is organized as follows: Section 2 describes related works related to deformations in CAOS systems. Following that, in Section 3, we briefly explain the MLS deformation technique. Section 4 explains the proposed deformation method in detail. Next, in Section 5, experiments and results are presented. Finally, conclusions and future work are presented in Section 6.

2. Related Work

Michalíková et al. [MBP*10] define the digital preoperative planning process as fast, precise and cost-efficient. Its main goal is to improve overall surgical performance and thus patient outcomes. Furthermore, it provides a permanent archived record of the templating process. A few medical areas require preoperative planning as essential part of daily practice.

CAOS systems are widely used in several studies and clinical trials for hip, spinal, knee, trauma and tumor surgeries, preplanning and simulation. For instance, a notable case of CAOS system is presented by Friederich and Verdonk [FV08], which is used for total knee replacement. This work shows the importance of CAOS systems as a useful tool for improving the alignment of prosthesis. Ollé et al. [OEK*06] also developed a system for preoperative planning, based on images of fractured bones. In their work, surgeons can insert implants while performing a virtual operation on a 3D model of the patient's bone. In this process, they can also join broken bone parts as if they were pieces of a puzzle. These two works require the input of Computed Tomography (CT) scans. However, not all preoperative planning systems work only with 3D images. Some CAOS systems also work with 2D images. An example of this is the work of Steinberg et al. [SSMD10] which describes a successfully preoperative planning of total hip replacement using 2D X-Ray images. Jamali [Jam09] also presents a preoperative surgical planning using standard radiographies and 2D implant templates.

Sometimes, when performing a surgery over a patient requiring an implant, this needs to be bended for it to fit correctly into the patient's anatomy. According to Korner et al. [KLM*03] fractures around the joints are common clinical cases where a patient requires a bended implant. During clinical practice, surgeons invest time in this process and sometimes they repeat it several times until they can finally reduce the fracture correctly. A relevant work in that area was presented by Sagbo et al. [SMDV*05]. They implemented several classical algorithms to bend 3D osteosynthesis plates for 3D preoperative planning of orthopedic surgery.

Nevertheless, such deformations can also be done for preoperative planning system which works with 2D images. Several algorithms have been developed for 2D image deformation. An important contribution in this area was introduced by Schaefer et al. [SMW06], which proposed a 2D image deformation based on linear MLS. Their work is an improvement of the work presented by Igarashi et al. [IMH05], which used a large linear equation system to make the deformation, while Schaefer et al. [SMW06] requires only a small $2x2$ linear system to accomplish deformation.

In the following sections we present a new method for 2D implant deformation which achieves very similar results to the real bending in the surgery room. The presented method is a new variant of the work of Schaefer et al. [SMW06], specifically applied to the problem of bending implant for preoperative planning for fractures of the lower extremities. Before explaining our approach, a brief background on the MLS deformation method is given in the next section.

3. MLS Transformation

Alexa et al. [ACOL00] introduced the concept of *as-rigid-as-possible transformation*, which consists on a rigidity-preserving interpolation in the form of a quadratic minimization problem. Using this transformation, it is possible to deform a model with minimum distortion during the process. This property allows translating and rotating the model without scaling and shearing it. Schaefer et al. [SMW06] applied the as-rigid-as-possible concept to a Moving Least Squares (MLS) transformation for 2D image deformation. This section describes the MLS transformation closely following the work of Schaefer et al. [SMW06]. The following sections propose changes in this model and explains how we apply the MLS transformation in our CAOS system.

The goal of the MLS transformation is minimizing the least square error function obtained through the mapping transformation process. In this process, a set of handlers (control points) are located inside the model that is going to be deformed and a transformation function is obtained for each point in the model. This function corresponds to a weighted least square function evaluated at each point of the model. The weight ensures that the effect of a control point is seen mostly in the zones immediately around it, while its effect is lessen in far zones.

When using MLS for image deformation it is possible to find, for a given point v inside the source image, the best affine transformation $l_v(x)$ that solves the following equation:

$$\min_{l_v} \sum_i w_i |l_v(p_i) - q_i|^2 \tag{1}$$

where each p_i corresponds to an initial handler, each q_i corresponds to the same handler after being modified by the user and each w_i is a weight calculated as follows:

$$w_i = \frac{1}{|p_i - v|^{2\alpha}} \tag{2}$$

where α represents a decay constant. Since weights w_i are dependent on the point v a different transformation $l_v(x)$ is obtained for each v.

Since $l_v(x)$ is an affine transformation, it can be written as

$l_v(x) = xM + T$, where M is a linear transformation matrix and T is a translation vector. After this, it is possible to eliminate translation T and rewrite the least squares problem of Eq. 1 as follows:

$$\sum_i w_i |\hat{p}_i M - \hat{q}_i|^2 \tag{3}$$

where $\hat{p}_i = p_i - p_*$ and $\hat{q}_i = q_i - q_*$. The term p_* corresponds to the weighted average of all p_i points, i.e., $p_* = \frac{\sum w_i p_i}{\sum w_i}$ and q_* corresponds to the weighted average of all q_i points, i.e., $q_* = \frac{\sum w_i q_i}{\sum w_i}$.

Several variations of M are possible. According to Schaefer et al. [SMW06], three main different classes of transformation are distinguished: *affine*, *similarity* and *rigid*. For the bending process of our CAOS system, we chose a rigid transformation because it preserves image proportions the most among all three classes of transformation. The next subsection describes the rigid transformation.

3.1. Rigid Transformation

Affine transformation matrices define non-uniform scaling and shearing, making the visual results undesirable for our CAOS system. Similarity transformation matrices are a special subset of affine transformations which include translation, rotation and uniform scaling, without shearing. To accomplish this, a constraint is applied over matrix M, which is that it must satisfy that $M^T M = \lambda^2 I$ for some scalar λ.

Now, if matrix M can be defined as a block matrix of the form:

$$M = (M_1\ M_2) \tag{4}$$

where M_1 and M_2 are column vectors of length 2, then restricting M to be a similarity transform requires that $M_1^T M_2 = 0$, which implies that $M_2 = M_1^\perp$ where the operator $\perp$ corresponds to the operation $(x,y)^\perp = (-y,x)$.

Although this restriction has been introduced, the minimization problem from Eq. 3 is still quadratic in M_1 and can be rewritten as finding the column vector M_1 that minimizes the following:

$$\sum_i wi \left| \begin{pmatrix} \hat{p}_i \\ -\hat{p}_i^\perp \end{pmatrix} M_1 - \hat{q}_i^T \right|^2 \tag{5}$$

Then, the solution for a similarity transformation function can be computed as follows:

$$f_s(v) = \sum_i \hat{q}_i \left(\frac{1}{\sum_i w_i \hat{p}_i \hat{p}_i^T} A_i \right) + q_*$$

$$A_i = w_i \begin{pmatrix} \hat{p}_i \\ \hat{p}_i^\perp \end{pmatrix} \begin{pmatrix} v - p_* \\ -(v - p_*)^\perp \end{pmatrix}^T \tag{6}$$

Similarity transformations preserve angles on images better than affine transformation. However, it allows local scaling which can often lead to undesirable deformations. In order to avoid this, a rigid transformation must be applied.

The matrix for the rigid transformation can be obtained by eliminating the uniform scaling, i.e. the scaling constants, from the similarity transformation matrix. The solution is simple and it has a closed form. It can be obtained easily by a slight modification of the similarity transformation for which the transformation matrix must satisfy the condition $M_1^T M_2 = M_1 M_2^T = \lambda^2 I$.

The rigid deformation vector $\vec{f_r}(v)$ is a rotated and scaled version of vector $v - p_*$, defined as follows:

$$\vec{f_r}(v) = \sum_i \hat{q}_i A_i \tag{7}$$

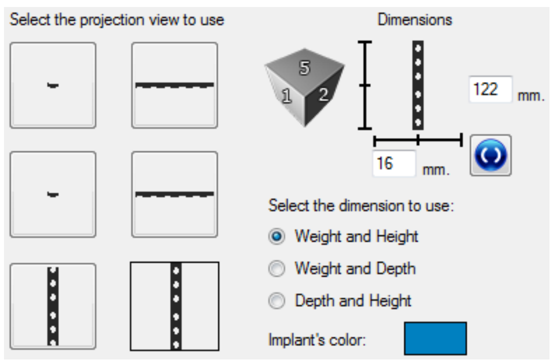
Matrix A_i is the same used in Eq. 6. Now, to compute the function $f_r(v)$ the vector $\vec{f_r}(v)$ should be normalized scaling it by $\mid v - p_* \mid$, and translating it by q_*, as follows:

$$f_r(v) = |v - p_*| \frac{\vec{f_r}(v)}{\mid \vec{f_r}(v) \mid} + q_* \tag{8}$$

This is the rigid transformation function for a point v inside the image using a MLS technique. The following section describes how this concept is applied over the implants used in our CAOS system.

4. Implant Deformation

In our CAOS system, implants are stored in STL format. The user selects the implant from a database and then the corresponding STL file is loaded. The database also contains the height, width and depth of each implant. Since the STL file contains the implant in 3D and our system uses X-Rays, the implant model must be projected to 2D before being overlaid onto the X-Ray. The user is presented a dialog box where he can choose the correct projection for the current planning, from among six possible axis-aligned orthogonal projections, see Figure 1. Although arbitrary projections might be required for very unusual cases, for most clinical cases using one of these six possible orthogonal projections suffices.


Select the projection view to use

Dimensions

122 mm.

16 mm.

Select the dimension to use:

[✓ Weight and Height]

Weight and Depth

Depth and Height

Implant's color: 🟦

Figure 1: *Implant loaded from database and the six possible projections to be selected.*

4.1. Deformation Handlers

Once the implant is placed in the preoperative planning, it can be translated or rotated. The bending process is performed using handlers placed along the implant. In Figure 2 these handlers are shown as small red squares. Now, according to Schaefer et al. [SMW06] using these handlers alone for the deformation might cause a foldback effect, because they could move the handlers freely. In our solution, this effect is undesirable because it distorts the implant as well as it changes its proportions. In order to avoid that, an OBB (*Oriented Bounding Box*) is constructed using the implant to constraint the movement of the handlers. Then, handler movement is only possible in a direction perpendicular to the major OBB axis. In Figure 2, this movement restriction is indicated by the direction of the red arrows.

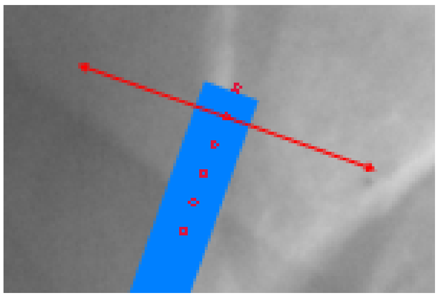


Figure 2: *Blue implant overlaid onto X-Ray. The small red squares correspond to deformation handlers. The two red arrows indicate the directions in which the handlers can be moved.*

4.2. Deformation Grid

On each step of our algorithm, we compute function $f_r(v)$ and recalculate all variables depending of q_i. Calculating function $f_r(v)$ for each pixel in the source image is too expensive in terms of execution time. Therefore, we overlay onto the implant a uniform grid of $m \times n$ squares, with $m+1 \times n+1$ vertexes, hereafter referred to as the *deformation grid*. The distance between vertexes d is constant, i.e. a vertex $v_{x,y}$ is d pixels away from neighboring vertexes $v_{x\pm d,y}$ and $v_{x,y\pm d}$. With this grid, the deformation can be computed per vertex instead of per pixel, therefore reducing computation time and accelerating the visual feedback. Figure 3 shows three different resolutions for the deformation grid, corresponding to different values of d.

The execution time of the algorithm is directly proportional to the number of vertexes in the deformation grid. Furthermore, the quality of the deformed image improves as d decreases, and it worsens as d increases. This is because the algorithm has to approximate more pixel values using bilinear interpolation.

We found that a value of d equals to the 5% of the largest image dimension is enough to obtain a deformed image with a good quality. Note that if this value is too small, e.g. $d = 2$, then it is not worthy to use the deformation grid since the execution time would be very close to calculate the deformation per pixel. However, this is not the case for implant plates for fractures of the lower extremity which generally are larger than 200 pixels.

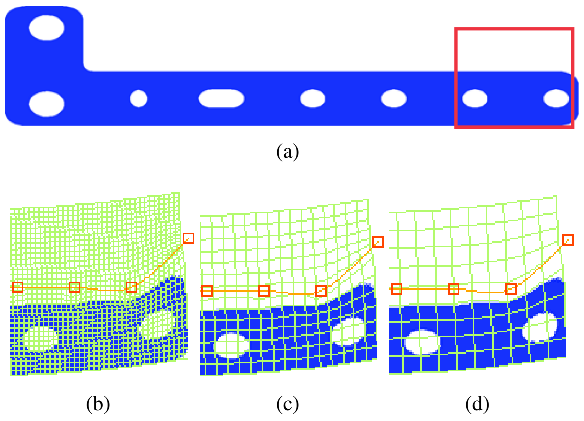


Figure 3: *Deformation grids of different resolutions over a zoomed area of the implant. (a) Original implant with a red square indicating the area of study (a) with $d = 2$, (c) with $d = 6$ and (c) with $d = 10$.*

4.3. Handler Distribution

For our specific case of implant deformation for fractures of lower extremities, the majority of handlers should be placed along the location where the deformation is going to be performed. Figure 4 shows an example where an implant with 10 handlers is shown, where the majority of handlers have been placed at the right end of the implant. In the figure, if handler B is moved, the deformation does not affect large parts of the image, but only a small area around the handler. This is because the deformation triggered by handler B is bounded by its directly neighboring handlers, which are not moved and are very close to B. Instead, if handler A is moved, the deformation affects all the area between its directly neighboring handlers, which are separated by a considerable distance. Then, a large part of the implant is affected by the deformation. Therefore, the majority of handlers should be placed along the location where the deformation is going to be performed, so that the surgeon can manipulate this area with more precision. By the majority of handlers, we refer to $60-70\%$. of the total number of handlers used.

In Schaefer et al. [SMW06] handlers must be placed manually by the user. In our CAOS system this can be done automatically. First of all, handlers are automatically placed along one of the lines in the deformation grid. Actually, the most centered line, parallel to the implant's major axis. In addition, the majority of handlers are placed in one of the following ways:

1. At either one of the ends of the implant.
2. Along both ends of the implant.
3. Along the central part of the implant.
4. Uniformly along the complete major axis of the implant.

For the first option, the majority of handlers are placed within the first k mm. from the selected implant end, where k should correspond to the $25-35\%$ of the implant's major axis length. In Figure 4, for instance, 7 out of 10 handlers have been placed in an area corresponding to 30% of the implant length. The rest of the handlers have been evenly distributed along the other 70% of the implant length.

A similar criterion is applied for the second and third options. For the second option, half the majority of handlers are placed within the first k mm. from one end of the implant, and the other half is placed within the first k mm. from the other implant end. For the third option, the majority of handlers are placed within an area of k mm. centered at the middle on the implant major axis.

The last option is provided for those cases in which the deformation must be performed in a place different than the ends or the middle of the implant. These cases are unusual, but they could certainly occur. In such a case, the handlers are simply placed along the major axis of the implant, separated by the same distance.

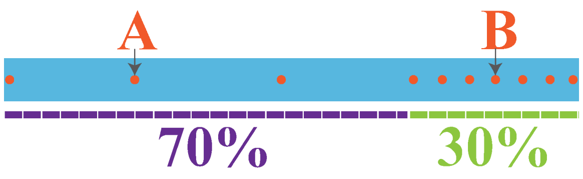


Figure 4: *An example of handler distribution along an implant. 7 handlers are evenly distributed in an area corresponding to* 30% *of the total implant length. The other 3 handlers are evenly distributed in the rest of the implant, corresponding to* 70% *of the implant length.*

4.4. Weight Function

The MLS deformation technique as presented by Schaefer et al. [SMW06] defines a weight function, shown in Eq. 2. Although this function definitely produces an image deformation, it is not suitable for our CAOS system, since the deformation produced with this function is not similar enough to the implant deformation in the real world. After numerous tests and the expert feedback of the Radiology Department at the University Hospital of Caracas, we found that a more suitable function was the inverse of Minkowski's distance function [Web99], obtaining the following equation for calculating the weight:

$$w_i = \frac{1}{D_{Minkowski(p_i,v)}} = \frac{1}{(\sum_{l=1}^{n} \mid p_{il} - v_l \mid^k)^{\frac{1}{k}}} \tag{9}$$

where p_i formed by (p_{ix}, p_{iy}) corresponds to an initial handler, v indicates a vertex of the grid and k represents the order of the function. With Eq. 9, we obtain better visual results than with the original function for w_i, because the inverse of Minkowski's distance function allows for pixels closer to a handler to be affected by the deformation more than pixels farther away from it. This function provides good results for the rigid deformation and does not distort implant holes.

4.5. Algorithm Details

We implemented a new variant of the MLS deformation technique. The algorithm is divided in four stages:

1. *Initialization*: Creates all necessary data structures used in the algorithm.
2. *Precalculation*: Calculates all constant values used during the MLS deformation, i.e., p_*, A_i and $\hat{p}_i$.
3. *Update*: When the user modifies the handlers to perform the deformation, the points q_i shown in Eq. 1 are modified. This stage is responsible for updating these values.
4. *Render*: Displays the deformed image.

The first two stages are executed only once. The last two stages are executed every time the user moves a handler. The resolution of the deformation grid is automatically calculated after the implant model is loaded and it cannot be changed throughout the deformation. The user can decide the number of handlers. The minimum number of handlers m that can be placed is $m = 5$. The maximum number of handlers h that can be placed is $m = \frac{w}{20}$, where w is the length of the implant's major axis. With this value the distance between two handlers represents the 5% of the implant's length.

The new MLS deformation variant was implemented in C++ with a GUI developed in C#. All implant images have a maximum resolution of 700×104 pixels. X-Ray images used in our preoperative planning are fixed to a maximum resolution of 1024×768 pixels. In Figure 5 we show an example of a complete preoperative planning made by a surgeon. The fractured bone in the figure is a femur. The clinical case was classified as 32B3 according to the *AO* fracture classification scheme [RBM07]. The surgeon used one 10-hole DCP (*Dynamic Compression Plate*) implant of 4.5 mm. and eight screws. Note that the DCP implant was deformed at one end, near the femur joint.

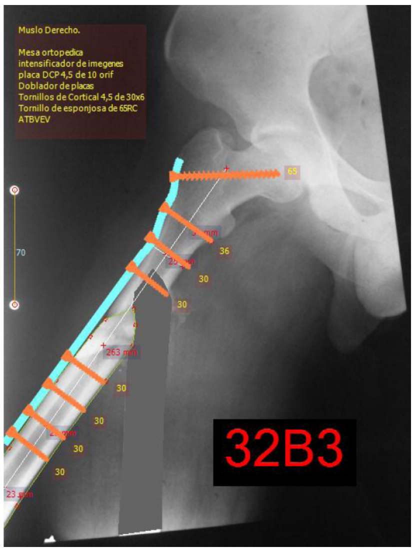


Figure 5: *An example of a complete preoperative planning using our CAOS system. One 10-hole DCP implant of* 4.5 *mm. is used (cyan). The planning includes 8 screws (orange).*

5. Tests and Results

For testing our new MLS deformation variant we simply performed several tests with different implants, varying the parameters of our techniques or simply removing them from the process, and then evaluating the visual results.

The first test focused on our proposal of restricting the movement of the deformation handlers using an OBB. As explained before, we restrict the movement of the handlers to two possible directions, both perpendiculars to the direction of the implant's OBB. Figure 6(a) shows a simple deformation example using our approach. Figure 6(b) shows the same implant deformation but without using the proposed restriction. It can be seen in this figure that the free movement of the handlers caused an undesirable distortion at the end of the implant. With our approach this effect is avoided. The restriction also makes the deformation process easier for the user.

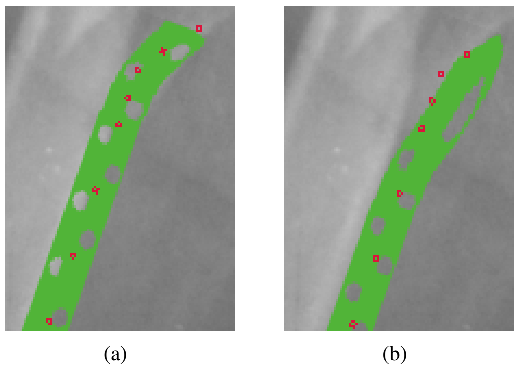


Figure 6: *Deformation handlers placed in an implant. If the movement is restricted then the preoperative planner can deform without errors. (a) Deformation applying the constraint (b) Deformation with free movement of the handlers.*

The second test was performed over our automatic deformation grid resolution calculation. As mentioned before, we use a value of $d = 5\%$ of the larger image dimension to define the resolution of the deformation grid. We tried values of d over 5% and found that such values produced noticeable distortion on the implant holes, see Figure 7(b). Also, it can be seen on the same image that the handlers are placed too far away from the implant. This is due to the large resolution of the grid, which produces large grid squares, and therefore when the handlers are placed onto the most centered grid line, they are place too far away from the implant. With our chosen value of d, the handlers are placed successfully close to the implant and the implant holes are not distorted by the deformation, see Figure 7(a). As explained in Section 4.2 it is not worthy to use values of d below 5%.

The third test evaluated the weight function to be used in the MLS. In this test we compared the obtained visual results using different weight functions. First, we use the original function for w_i, see Eq. 2. Figure 8(a) shows the results of this with $\alpha = 0.5$ where pixels are little affected by the deformation. The holes of the implant have a circular shape. In Figure 8(b) we used $\alpha = 1.0$. In this case the result exhibits a smooth deformation on the implant borders as well as a slightly deformed hole. Finally, in Figure 8(c) we used $\alpha = 1.5$. Here the deformation is more adjusted to reality, but it changes the proportions of the implant around the handler.

Note that when the value of α approaches zero, pixels are less affected by the deformation. In such a case, the movement of the handlers should be greater to achieve a more realistic deformation. On the other hand, as the value of α increases the deformation becomes more flexible and deforms the original proportions of the implant. To avoid that, we proposed a new weight function. Figure 9 presents the same deformation shown is Figure 8 but using our proposed weight

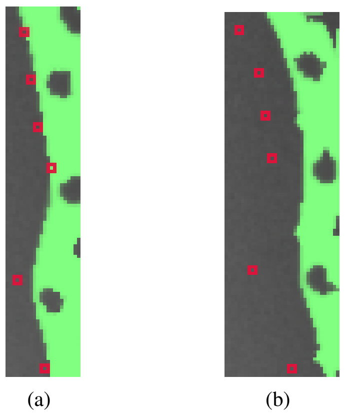


Figure 7: *A deformation example under different grid resolutions. The figure represents a small area for an implant with dimensions* 29×203 *pixels, (a) with grid resolution* $d = 10$ *and (b) with grid resolution* $d = 30$.

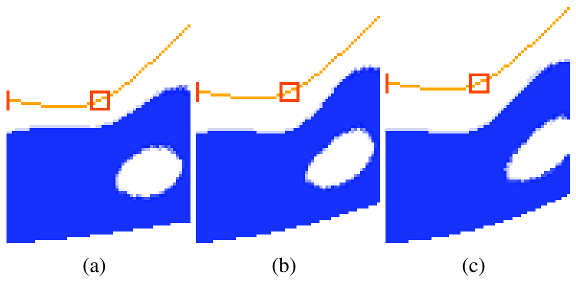


Figure 8: *The zoomed part of the implant in Figure 3(a) after a deformation. The original weight function was applied with different values (a)* $\alpha = 0.5$*, (b)* $\alpha = 1.0$ *and (c)* $\alpha = 1.5$.

function. Figure 9(a) shows the deformation with $k = 0.5$. This figure shows that pixels close to the handler are distorted and do not correspond to reality. Figure 9(b) uses $k = 1.0$, which obtains an acceptable result, except that the area of the implant below the hole is not deformed correctly. Finally, we tested our approach with $k = 2.5$, where the visual result is completely acceptable for our CAOS system. Figure 9(c) shows an example of this. In this last case, the deformation is performed without distorting the hole and it produces very similar results to the deformation performed in the surgery room.

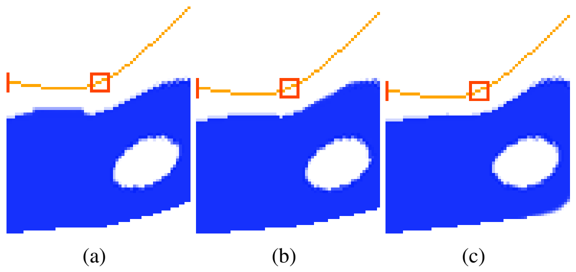


Figure 9: *The zoomed part of the implant in Figure 3(a) after a deformation. Our weight function was applied with different values (a)* $k = 0.5$*, (b)* $k = 1.0$ *and (c)* $k = 2.5$.

6. Conclusions and Future Work

In this paper, we have presented a new variant of the MLS deformation technique specifically for the bending of implant models for fractures of lower extremities. Our system has been tested by the members of the Radiology Department of the University Hospital of Caracas for the planning of over 100 clinical cases. According to them, our deformation technique achieves very similar visual results to the real bending of implants in surgeries.

We have shown that restricting the movement of deformation handlers using an OBB avoids possible errors caused by the surgeon. Moreover, we have shown that the handlers can be placed and distributed automatically along the implant model. The free movement of handlers might cause several distortions and it changes the proportion of implants. This is totally undesirable for our CAOS system.

We have also shown that the deformation grid resolution can be computed automatically for each implant model. In addition, we found that setting the size of each grid square to 5% of the implant's larger dimension is an adequate value to configure grid resolution.

This paper also introduced a new weight function for the MLS deformation technique. When compared with the function used in the original MLS formulation, the new function achieves more realistic visual result for our CAOS system. Moreover, this paper shows that to obtain such results a value of k greater or equal than 2.5 should suffice.

In the future, we are planning to improve the deformation by introducing a multiresolution deformation grid using a quadtree data structure. In Figure 10, we show an example of this. The multiresolution grid would allow the deformation of specific implant areas with more detail. The idea would be to have more resolution in those areas where a color change is detected, i.e., the borders and holes of the implant.

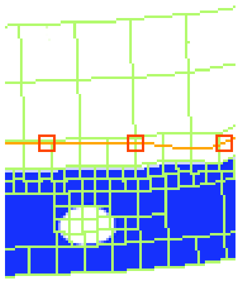


Figure 10: *Possible approach using a multiresolution deformation grid.*

Another possible future work would be to allow a handler some influence over its immediate neighboring handlers. In that way, when a handler is moved in a direction, the immediate neighboring handlers would also move in the same direction in some proportion, according to a predefined heuristic. This would speed up the deformation process in the planning.

Acknowledgments

This work was funded by the Scientific and Humanistic Development Council (CDCH) of the UCV. Also, we would like to thank Md. Carlos Sánchez, Eng. Othman Falcón and the members of the Department of Traumatology and Orthopedics of the University Hospital of Caracas (HUC), for their invaluable help in testing our system. All medical images used during this research were provided by the HUC.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 157–164. 2

[BJ03] BIRKHOLZ H., JACKÈL D.: Image warping with feature curves. In *SCCG '03: Proceedings of the 19th Spring Conference on Computer Graphics* (2003), ACM, pp. 199–202. 1

[FV08] FRIEDERICH N., VERDONK R.: The use of computer-assisted orthopedic surgery for total knee replacement in daily practice: a survey among ESSKA/SGO-SSO members. *Knee Surg Sports Traumatol Arthrosc* (2008). 2

[IMH05] IGARASHI T., MOSCOVICH T., HUGHES J. F.: As-rigid-as-possible shape manipulation. In *SIGGRAPH '05: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2005), ACM, pp. 1134–1141. 2

[Jam09] JAMALI A.: Digital templating and preoperative deformity analysis with standard imaging software. *Clinical Orthopaedics and Related Research 467* (2009), 2695–2704. 2

[KLM*03] KORNER J., LILL H., MÜLLER L. P., ROMMENS P. M., SCHNEIDER E., LINKE B.: The LCP-concept in the operative treatment of distal humerus fractures–biological, biomechanical and surgical aspects. *Injury 34* (2003), B20–30. 2

[MBP*10] MICHALIKOVA M., BEDNARCIKOVA L., PETRIK M., ZIVCAK J., RASI R.: The digital preoperative planning of total hip arthroplasty. *Acta Polytechnica Hungarica 7*, 3 (2010). 2

[OEK*06] OLLÉ K., ERDÖHELYI B., KUBA A., HALMAI C., VARGA E.: MedEdit: a computer assisted image processing and navigation system for orthopedic trauma surgery. *Acta Cybernetica 17*, 3 (2006), 589–603. 2

[RBM07] RÜEDI T. P., BUCKLEY R. E., MORAN C. G.: *AO Principles of Fracture Management*, 2nd ed., vol. 1. Thieme Medical Publishers, 2007. 5

[RC10] RAMÍREZ E., COTO E.: Digital preoperative planning for long-bone fractures. In *Modelos Computacionales en Ingeniería: Desarrollos Novedosos y Aplicaciones* (2010), Sociedad Venezolana de Métodos Numéricos en Ingeniería, pp. TC 1–6. 1

[SMDV*05] SAGBO S., MARTI G., DI VENUTO C., VANGENOT C., NOLTE L., ZHENG G.: Design and implementation of deformation algorithms for computer assisted orthopedic surgery: application to virtual implant database and preliminary results. *Engineering in Medicine and Biology Society 7*, 1 (2005), 6946–6949. 2

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. In *SIGGRAPH '06: Proceedings of the 30th annual conference on Computer graphics and interactive techniques* (2006), ACM, pp. 533–540. 2, 3, 4, 5

[SSK07] STANNARD J. P., SCHMIDT A. H., KREGOR P. J.: *Surgical Treatment of Orthopaedic Trauma.* Thieme Medical Publishers, Inc., 2007. 1

[SSMD10] STEINBERG E., SHASHA N., MENAHEM A., DEKEL S.: Preoperative planning of total hip replacement using the TraumaCad system. *Archives of Orthopaedic and Trauma Surgery 130* (2010), 1429–1432. 2

[Web99] WEBB A.: *Statistical pattern recognition.* Arnold Publishers, 1999. 5

[YSTS*09] YUSOF S. F., SULAIMAN R., THIAN SENG L., MOHD. KASSIM A. Y., ABDULLAH S., YUSOF S., OMAR M., ABDUL HAMID H.: Development of total knee replacement digital templating software. In *Proceedings of the 1st International Visual Informatics Conference on Visual Informatics: Bridging Research and Practice* (2009), Springer-Verlag, pp. 180–190. 1