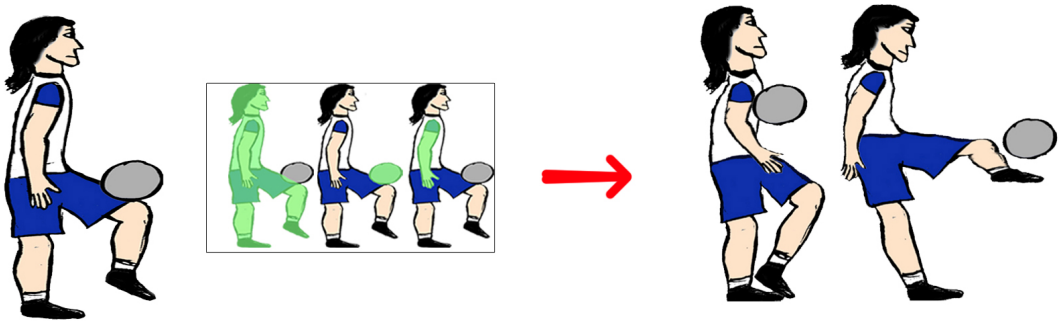
2D Shape Deformation Based on Positional Constraints and Layer Manipulation

Tiago Mota
PESC/UFRJ

Claudio Esperança
PESC/UFRJ

Antonio Oliveira
PESC/UFRJ



Abstract

This paper presents an interactive system for deforming two-dimensional objects based the manipulation of positional constraints and the creation and editing of layers representing parts of the object. The layers may be deformed in a independent way, or they can be related through regions of interest chosen by the user. Using tools for editing and deforming layers it is possible to achieve more natural results than single-layer approaches. The system is particularly suited to the animation of characters in the form of cartoons where arms or legs overlap the body or each other.

Keywords: shape deformation, position constraints , 2D animation, layers

Author's Contact:

tiagosmota@gmail.com
esperanc@cos.ufrj.br
oliveira@cos.ufrj.br

1 Introduction

The study of two-dimensional deformation aims to provide efficient methods for modeling and shaping 2D objects embedded in images. These methods proved to be very important in many applications, such as enrichment of graphical interfaces [Thomas and Calder 1995], character animation [Sýkora et al. 2009]and image editing. Techniques for 2D deformation are particularly appealing when applied in a character animation context, enabling the artist to obtain several poses for the same character without redrawing. Fast controlled deformations are also useful in interactive games for mimicking collision response of objects or characters.

When creating a image deformation system, a number of factors should be considered:

- *Performance*. In most cases the goal is to have an interactive system that generates real-time results from a series of simple operations.
- The *nature of the object* to be deformed. The way that these objects behave in the real world should be respected when choosing the method to be used for its deformation. In other words, the result of the deformation should be physically plausible.
- *Consistency* of results. The technique should allow little margin for error or noise. It is not desirable to use a system that does not generate visually pleasing results.
- *Interface*. An intuitive interface allows greater efficiency in using the system.

Most methods for two-dimensional shape deformation have their origin in the study of three-dimensional object deformations. Although the properties to preserve are not always the same, the expected results are very similar. Thus, it is common to find references to works about 3D shape manipulation in papers about 2D shape deformation.

While several paradigms for the deformation of images have been proposed in the literature, some of the most popular are collectively known as free-form deformation (FFD) as can be seen, for instance, in the work of [MacCracken and Joy 1996]. Methods based on this paradigm can achieve good results by partitioning the the domain of the image and they obtain their results through the manipulation of a set of control points for each subdomain. These points can be moved freely thus deforming the corresponding parts of the domain. The problem with this approach is that the creation of subdomains and the definition of control points may be time-consuming, since the number of points can be very large. In addition, the FFD methods do not take into account the natural way in which objects move in the real world.

Deformation methods based on the manipulation of skeletons (e.g., [Yan et al. 2008]) are also popular. This approach is intuitive, but, as the name suggests, it is crucial to create a skeleton that represents the articulated characteristics of the object. The shape of the object is changed by moving the bones and joints of the skeleton created. These methods, however, do not achieve satisfactory results for objects that don't have an articulated structure, or can't be represented by one. In addition, to define a skeleton is not a trivial task and setting the weights for the joints can become quite tedious. Like the free-form deformation methods, those based on manipulating skeletons are computationally efficient and are easy to implement, but they don't have convenient tools for user interaction [Wang et al. 2008].

There are also methods of deformation that make use of physically-based simulations [Celniker and Gossard 1991]. These methods are also known as non-linear optimization methods. They often show very good results, but they are too slow for an interactive approach and have a very high computational cost.

This paper presents a new interaction approach to 2D shape deformation based on intuitive positional constraints, introduced in the work of [Igarashi et al. 2005]. They are similar to FFD methods and in the sense that deformation is achieved by selecting points in the object to function as positional constraints, and by moving them, the system performs a series of geometric operations to obtain a new form for the object. Our method uses the concepts of layer editing and image processing to increase the mentioned deformation paradigm, through an interactive and easy to use interface.

It must be noted that the mathematical framework for producing the deformation is not significantly changed by our approach. Rather, we show how several layers can be integrated in the framework pre-

sented by Igarashi. Other frameworks, such as the one presented by [Weng et al. 2006], could also be used. In this sense, the deformation machinery is treated as a "black box" in our system.

2 Related work

In recent years, several image deformation approaches based on the use of positional constraints have been proposed. These methods can achieve physically acceptable results for many different shapes by moving control points chosen by the user. Although they have different representations and linear algorithms, these methods have a number of common characteristics. In particular, they

- strive to maintain the shape of the inner regions of the objects;
- use polygon partitioning techniques;
- try to interpolate the positional constraints as they are moved;
- work at interactive performance rates;
- have intuitive interfaces.

A seminal work where image deformation is achieved using positional constraints was presented by [Igarashi et al. 2005]. By creating a 2D triangular mesh, the method allows interactive deformation of the object through the manipulation of control points, which are chosen by the user. Mathematically, the deformation process is accomplished during user interaction by two linear optimization steps. The first computes the changes in scaling, while the second computes rotation of the mesh triangles. The algorithm aims to find the configuration that minimizes the total distortion for all triangles.

A related work using positional constraints was developed by [Weng et al. 2006]. This work considered that not all components of the energy function representing the deformation are linear, and instead of trying a linear approximation of the terms of the energy function to be minimized, it uses iterative techniques to solve a nonlinear system. The method aims to preserve two important geometric properties of two-dimensional objects: the Laplacian coordinates of the curve limiting the object and the local areas within the object.

Another approach for image deformation with positional constraints was developed by [Schaefer et al. 2006], where the energy minimization is posed as a moving least squares (MLS) problem. The authors derive a closed formula involving only the constraint points which, and thus are able to deform the whole image. Unfortunately, the technique is blind to the actual shape of the objects embedded in the image. Later, this approach was extended in [Cuno Parari et al. 2009] to 3D domains and adapted to the deformation of meshes by means of a skeleton-guided scheme.

Several other approaches for image deformation which do not involve positional constraints have also been proposed. For instance, [Eitz et al. 2007] and [Pereira et al. 2011] describe methods employing hand-drawn sketches.

It should be emphasized, however, that the main contribution of this work in the use of layers in order to achieve a more controlled deformation of 2D objects. Thus, any deformation algorithm employing meshes to subdivide the object domain can be adapted to the proposed layer framework. The concept of layers can also be adapted to other schemes which do not use meshes, although, in this case, some other means for establishing the linkage between layers must be devised.

3 Solving the deformation problem

The interaction method developed in this work uses the numerical solution described in [Igarashi et al. 2005] to create a system of deformation based on the use of layers and image processing tools. The main goal is to allow a finer control over the deformation such that moving control points does not necessarily affect the whole object, but only local partitions – i.e., layers – which the user has the freedom to select.

3.1 Overview of the method with a single layer

It is educational to examine the original deformation method proposed by [Igarashi et al. 2005]. Our own system strives to reproduce the same functionality of that system and may be regarded as equivalent when a single layer is defined.

The first step is the creation of a polygon that wraps the object while preserving its silhouette. This can be done manually or, if the image has a good background to foreground contrast ratio, it is possible to use an automatic method, such as *marching squares* algorithm, which is a 2D version of the well-known *marching cubes* algorithm [Lorensen and Cline 1987].

After obtaining the polygon formed by the border points, obtain a triangulation of its interior. The work of [Igarashi et al. 2005] uses a triangular mesh generated by the method proposed by [Markosian et al. 1999]. In our implementation we use the triangulation method described in [Hemmer 2009], as implemented in the CGAL[1] library, due to its performance and detailed documentation. From this moment on, the user must choose which vertices of the triangular mesh will be used as positional constraints.

It should be noted that the partitioning of the object into triangles is crucial to the performance of the deformation system. A greater number of triangles yields a smoother deformation, but also means that a larger linear system must be solved during the movement of constraints.

As the triangulation algorithm uses the contour points as vertices, it is important that the set of points representing the contour do not become very large. Thus, it is useful to apply a simplification algorithm to reduce the number of points of the polygon without causing major damage to its shape. For this purpose we used the Douglas-Peucker [Douglas and Peucker 1973] algorithm to get a good representation for the polygon rapidly (see Figure 1.c).

The last step is to use the deformation method. The implementation of this method was made following the route described in [Igarashi and Igarashi 2009]. A full account of that method is out of the scope of the present paper. In summary, it consists of using the connectivity information of the mesh in order to build two least-squares problems. The first problem consists of obtaining similarity transformations for all triangles subject to the mesh topology and the constraints imposed on the vertices selected as control points. The second problem consists of adjusting the scaling of the triangles so that they do not enlarge or shrink too much. The bulk of the computation is done in a pre-processing stage, so that obtaining new shapes by moving constraint points can be done at interactive rates (see Figure 1d).

3.2 Layer structure

In many cases, the results obtained with the approach described above look unnatural. An obvious cause for this is the fact that the image is usually a projection of a real 3D object. Thus, two entirely separate parts of that object may be superimposed in the image, and will be deformed together by the method (see Figure 1e). Even when the projection is favorable, the deformation itself may cause two parts of the original image to overlap (see figure 2). This can make the choice of the portion to be displayed inadequate and often random. In [Weng et al. 2006] and [Igarashi et al. 2005] cope with this problem using a static display order of object subparts, but this strategy does not work well in all types of object, especially in areas with many overlapping regions.

Another limitation of this global approach to image deformation is the lack of a structure to control small parts of the object. In other words, the result of the changes in a single point is calculated based on weighted calculations involving the position of its neighboring points, making points with fewer neighbors more susceptible to deformations. Thus, for greater control over deformations, a careful assignment of weights in different parts of the object would be needed. For this, [Igarashi et al. 2005] propose a painting interface for manual assignment of weights at each vertex. This solution,

[1] http://www.cgal.org.

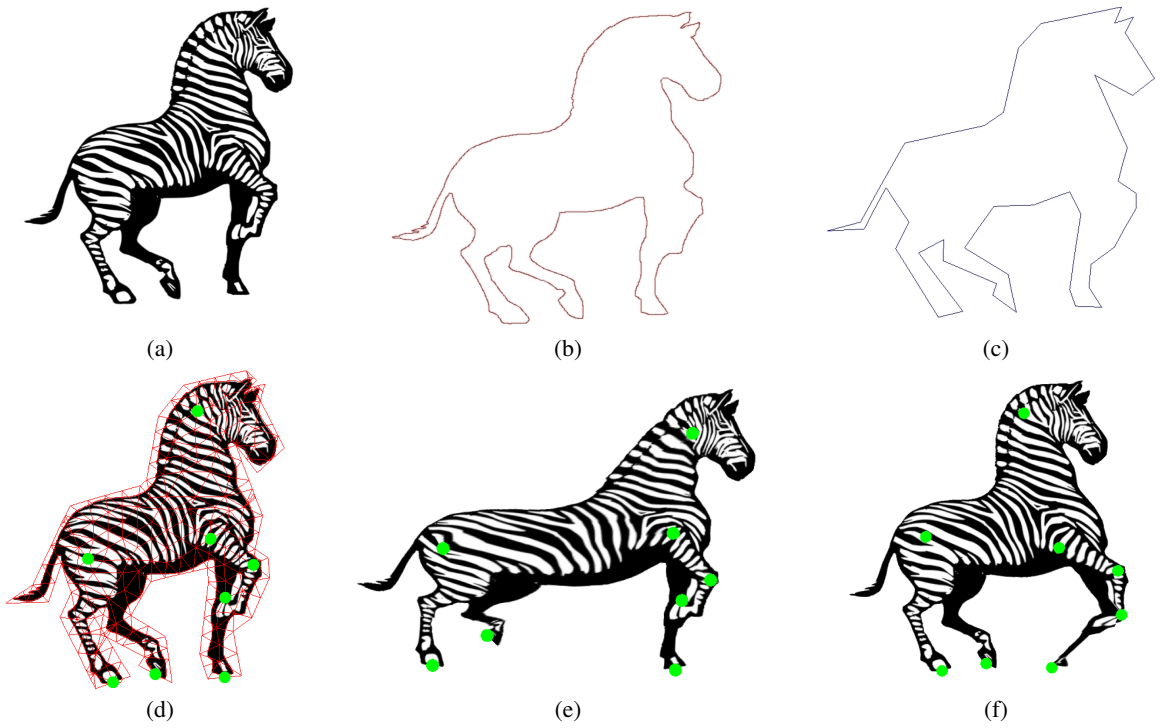


Figure 1: *Single layer algorithm: (a) Original image, (b) contour extraction, (c) simplified contour obtained by the Douglas-Peucker algorithm, (d) triangulated image with positional constraints at some vertices, (e) and (f) deformed image.*

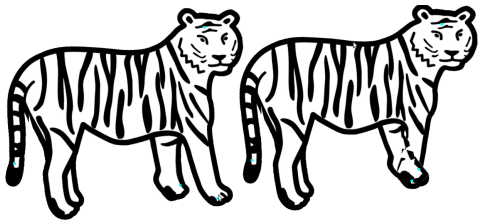


Figure 2: *Example of deformation in which the overlapping regions generates inconsistent results.*

however, besides requiring the animator to carefully edit the mesh in a non-intuitive way, does not address the problem of overlapping parts.

To address these limitations, the present work makes use of a structure of layers, which represent different and possibly overlapping regions of the object. The structure is based on the selection of portions of the object, so that they begin to share some characteristics, such as rigidity of the points, the display order of the segments, the influence of control points, among others (see Figure 3).

The delimitation of image parts that compose a given layer can be made based on metrics such as the existence of common characteristics between points, the existence of a real separation in the object or any other need to differentiate between points. As shall be seen in Section 4, in our system, the desired portions of the image are selected by the user with lasso or rectangle tools, painting and other image processing operations.

Once layers are created, their stacking order must be defined. By default, are stacked in order of creation, but this index can be changed at any time during the interaction.

After this step, the relationship between layers must be established. Sometimes, layers are to be deformed independenty. In this case, each layer would go through all steps described in Section 3.1. In some other cases, two overlapping layers may be attached to each other using one or more edges of the triangulation. These linkage edges are provided by the user by drawing one or more line segments inside the area of intersection between the layers. This is done in a step preceding the triangulation. Thus, although layer triangulations are largely independent, they are viewed by the deformation algorithm as a single triangulation.

4 Interaction

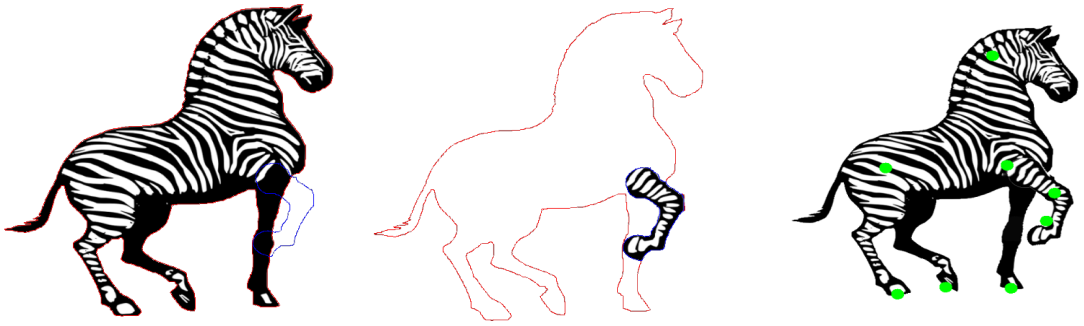
The application interface uses a modal paradigm which reflects the series of steps the animator would have to follow to obtain the poses for each character. In other words, the behavior of the mouse and the operations made available by the interface depend on which step of the deformation pipeline is being authored at the moment. To aid the user, a "task script" is shown as a series of buttons (see panel (1) in Figure 4). These tasks are summarized below.

4.1 Image loading

Initially, the user selects the image containing the object to be deformed. Any images or layers previously defined are erased. Once loaded, the image is shown in the central area of the interface and the first layer is created (panel (5)). At this point, the program tries to guess which pixels belong to the object to be deformed by applying a simple brightness threshold segmentation algorithm. The result of this process is a bitmap mask where 1's correspond to object pixels and 0's indicate background pixels. In this stage, it is also possible to add or subtract to the mask using painting tools. At the end of this phase, the image is stored in a separate buffer to serve as a template for creating new layers.

4.2 Layer editing

This is probably the most time-consuming part of the interaction. Layers can be added or removed using the "+" and "−" buttons in the layer panel (panel (5)). By clicking on the name of the layer, this layer is loaded into the central area so it can be modified. It is also possible to manipulate the stacking order of the layers by dragging names up and down in the layer panel. Other than the first layer, newly created layers are assigned empty bitmap masks. Pixels from the template buffer created in the initial step can be cloned into a new layer by using cloning brushes or polygon cloning tools (panel (2)), meaning that corresponding bits in the bitmap mask are set to 1. Similarly, pixels of a layer can be removed by



(a) (b) (c)

Figure 3: *Example using layers: (a) The main body layer, (b) layer containing a single leg, and (c) sample deformation where legs can be separated.*

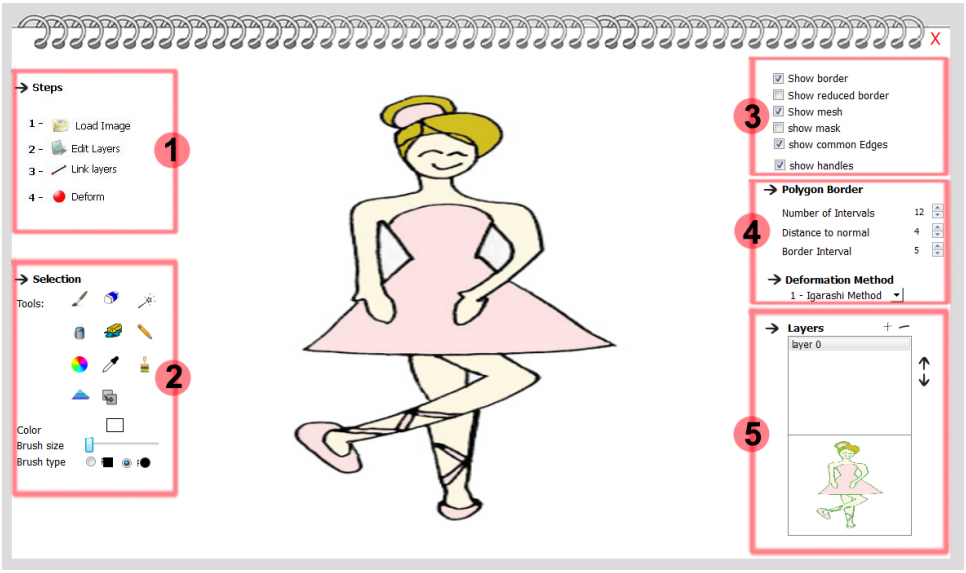


Figure 4: *Interface: (1) Task script; (2) Tools for editing and manipulating layers; (3) View options; (4) Triangulation and deformation options; (5) Preview and ordering of the layers.*

the erase tool. It is also possible to repaint areas of a layer using various kinds of brushes and colors (see example in Figure 5)

In our implementation, only a few simple tools are offered for retouching the visual aspect of each layer, but a separate dedicated image painting application could conceivably be used for this task. A key editing operation consists in retouching parts of a given layer which will be revealed as an overlapping layer is moved away from it. This is known as *inpainting*. Many techniques can be used to help the artist in this step. Our system implements only a simple pyramid-based algorithm [Ogden et al. 1985], even though this particular technique is best suited for filling out relatively small areas of the image based on the colors of nearby pixels. See Figure 6 for an example where pyramid based inpainting does not lead to good results.

4.3 Layer linking

This step aims to relate the layers with each other. Suppose that two layers are not to be deformed independently. Then, the user first selects them in the layers panel. The system will then compute their common area them by performing a bitwise "and" between their bitmap masks. Using the pencil tool, the user then draws one or more line segments inside the intersection area (see Figure 5.c). These will later become common edges in the corresponding layer triangulations.

It should be noted that the number and the length of the line segments have a bearing in the degree of "connectedness" between layers. Longer lines imply a greater dependence between layers.

In this stage, the user must also define constraint points with the constraint tool. Constraint points are handles which later will be dragged to obtain new poses of the object. Since each constraint point is attached to only one layer, the user must select it on the layer panel before using the constraint tool. In practice, one must place restrictions either in parts of the image which must remain "anchored" during the deformation, or attached to parts such as legs or arms one wishes to drag to obtain new poses.

4.4 Deformation

By clicking on the *Deformation* button in panel (1), the system builds all data structures necessary for performing the interactive deformation of the object.

Firsty, the bitmap masks indicating the object pixels of each layer are submitted to the *marching squares* algorithm in order to define the outline of the regions to be deformed. Since the algorithm is applied by a simple scan from top to bottom, it only outputs a single polygon for each layer. This means that the outline may contain pixels which are not part of the object (see Figure 7). Notice that the outline polygon delimits the domain which will be triangulated. In general, this does not represent a problem, since pixels corresponding to bits set to zero in the bitmap mask will be transparent during the deformation phase. These polygons are subsequently simplified using the Douglas-Peucker algorithm (see Section 3.1). The degree of simplification may be configured using controls in panel (4).

Next, a restricted Delaunay triangulation for each layer is obtained according to the method described in [Hemmer 2009]. Recall that

a restricted triangulation is obliged to contain pre-established edges and vertices. In particular, the layer linkage line segments, as well as those segments comprising the border polygons are included as edge restrictions, while constraint points are set as vertex restrictions. The density of the desired triangulation can also be defined through controls in panel (4).

Finally, one deformation problem is built for each set of connected layers. This entails the construction of two least-squares problems, each containing $O(n-m)$ variables, where n is the number of triangles in all connected triangulations and m is the number of control points attached to them. Notice that these problems are factored so as to permit the bulk of the computation to be done only once at a cost of roughly $O((n-m)^3)$. This means that the user usually has to wait a few seconds after he/she hits the *deformation* button before being able to interactively move the positional constraints.

The interactive deformation of the triangle meshes is done in $O(m(n-m))$, which is almost linear with respect to the size n of the mesh for small m. In this stage, the user can click and drag the previously defined control points to achieve the desired shape for the object (see figure 8).

5 Results and tests

The system was tested on a computer with 2GB of RAM and a Intel Core ™ 2 Quad Q8300 2.5GHz processor. The operating system was Ubuntu 9.10 and the program was compiled with gcc. The triangulation algorithm was provided by CGAL[2] and linear equation solvers were provided by the Eigen[3] library. Since the numerical issues of the deformation framework were already reported in [Igarashi et al. 2005], we see no reason to expand on this here. Suffice it to say that the system behaves at interactive rates ($>$ 20 fps) only for meshes having around 300 triangles.

The layer structure allows a more precise control over the objects. This is also due to the fact that the layers can be associated with triangular meshes with different levels of refinement. Thus, for thinner regions, which are generally more susceptible to deformation, can be made more rigid, which is more awkward to do in the non-layered approach (see Figure 9).

The relationship between layers is also a factor to be taken into account. When the layers are related to each other (see figure 10), the deformation process is more practical and fast, but in some cases it may be necessary to adjust the position of the common constraints to achieve better deformation for the object. Unrelated layers (see figure 11 and 13) tend to leave the process of deformation more loose, but may leave him more tedious.

The number of layers is another factor that may contribute to the complexity of the deformation setup. In general, a greater number of layers is needed for more flexible character posing, but this also entails that the user must spend more time preparing and retouching the layers, and these may have to be set into the desired positions using more constraint points. Several deformation examples obtained with our system are shown in Figures 10 to 13

[2] http://www.cgal.org
[3] http://eigen.tuxfamily.org

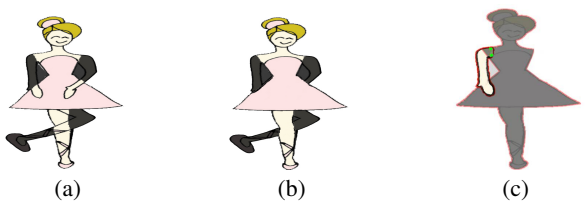


Figure 5: *Defining and linking layers: (a) Pixels from the template image are cloned into the layer; (b) pixels of the layer are painted to remove overlapping parts of another layer; (c) two layers a linked by a line segment.*

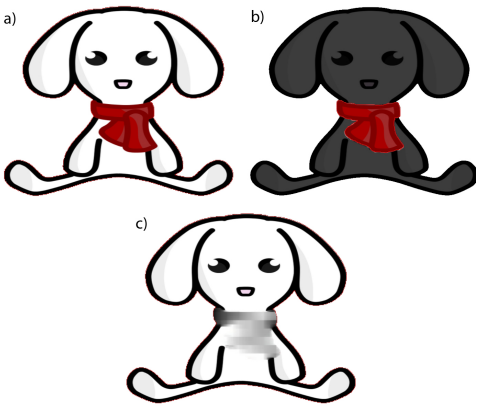


Figure 6: *Example of the use of pyramid-based inpainting algorithm in regions not too small.*

Figure 7: *Polygon delimiting the object outline.*

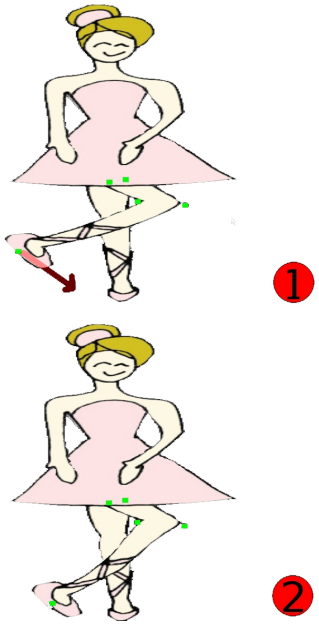


Figure 8: *constraints manipulation: 1- Position1; 2- Position 2.*

6 Limitations and future work

Many of the limitations of the described system are inherited from the mathematical framework used to deform one layer. The most significant ones are:

- The non-intuitive deformations achieved for some large displacements of control points. This is apparent, for instance, in the legs of the football player and the ballerina in certain poses of Figures 12 and 13. Some of these problems are reported in the original paper by Igarashi et al. They suggest altering the weights assigned to some vertices of the triangulation during the optimization process in order to increase

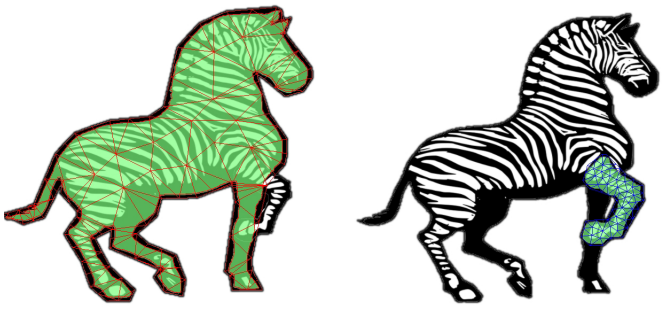


Figure 9: *Example of layers with different levels of refinement.*

the rigidity of the incident triangles. This change in the assignment of weights can be effected by means of a painting interface, which would fit nicely with the other painting tools in our system's interface. Another possibility is to employ another deformation scheme. Some natural candidates are the the non-linear optimization approach of [Weng et al. 2006] and the moving least-squares (MLS) approach of [Schaefer et al. 2006]. Another possibility is the adaptation of standard 3D deformation schemes based on skeletons and skins. The main challenge of these methods is that the skeleton must be built by hand, although some authors report automated methods for this (see, for instance, [Vasilakis and Fudos 2009]).

- The limitation on the density of triangulations which can be interactively deformed (currently, around 300 triangles). This limitation can be alleviated to some degree using better numerical solvers or offloading the bulk of the numerical workload to the GPU using GPGPU techniques. Obviously, the problem can also be tackled by using less computationally intensive deformation schemes such as MLS or skeleton-based approaches.

The introduction of layers and the proposed scheme for propagation of deformations across layers are also subject to some issues which deserve further investigation:

- The separation of an original drawing into layers may be somewhat time-consuming. In particular, the retouching of areas which are uncovered by moving away an overlapping layer may require artistic skills from the animator. This situation may be helped, for instance, by using a better algorithm for inpainting such as [Yamauchi et al. 2003].

- The linking between layers effected by drawing line segments offers limited control over the "rigidity" of the binding. There are quite a few alternative schemes which could be tried in this respect. For instance, one could admit linkage points or linkage polygons. Another idea would consist of deforming layers independently but feeding back deformed vertices as restrictions in other layers.

- The use of non-positional restrictions could also lead to added control to the authoring of deformations. For instance, [Schaefer et al. 2006] describe the use of line segments as restrictions.

7 Conclusion

In this work was made a study of methods based on the paradigm of image deformation based on the use of intuitive positional constraints. Our contribution to the work of this approach was the use of a layered structure that defines a different representation for the object and allows greater freedom in its deformation.

We realized that the use of layers is a viable option to add more possibilities to the deformation of images based on the use of positional constraints without considerably affecting the performance. In addition, the use of tools of images processing, often familiar to the user, permits a smooth learning curve.

Our experience in using the system for simple deformations suggests that many of the operations can be simplified and make more natural with a less cluttered graphical user interface. Also, since the intended use for the system is in cartoonish character animation, its integration into an animation framework is planned for the future.

References

CELNIKER, G., AND GOSSARD, D. 1991. Deformable curve and surface finite-elements for free-form shape design. *SIGGRAPH Comput. Graph. 25*, 257–266.

CUNO PARARI, A. E., ESPERAN CA, C., AND OLIVEIRA, A. A. F. 2009. Shape-sensitive mls deformation. *Vis. Comput. 25* (September), 911–922.

DOUGLAS, D. H., AND PEUCKER, T. K. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization 10*, 2 (Oct.), 112–122.

EITZ, M., SORKINE, O., AND ALEXA, M. 2007. Sketch based image deformation. In *Proceedings of Vision, Modeling and Visualization (VMV)*, 135–142.

HEMMER, M. 2009. *Polynomials, CGAL - Computational Geometry Algorithms Library, release 3.4*. CGAL, Campus E1 4, 66123 Saarbrücken, Germany, January.

IGARASHI, T., AND IGARASHI, Y. 2009. Implementing as-rigid-as-possible shape manipulation and surface flattening. *Journal of Graphics, Gpu, and Game Tools 14*, 1, 17–30.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, 1134–1141.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 163–169.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 181–188.

MARKOSIAN, L., COHEN, J. M., CRULLI, T., AND HUGHES, J. 1999. Skin: A constructive approach to modeling free-form shapes. In *Proceedings of SIGGRAPH 99*, 393–400.

NGO, T., CUTRELL, D., DANA, J., DONALD, B., LOEB, L., AND ZHU, S. 2000. Accessible animation and customizable graphics via simplicial configuration modeling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 403–410.

OGDEN, J. M., ADELSON, E. H., BERGEN, J. R., AND BURT, P. J. 1985. Pyramid-based computer graphics. *RCA Engineer 5*, 4–15.

PEREIRA, T., VITAL BRAZIL, E., MACÊDO, I., DE FIGUEIREDO, L. H., AND VELHO, L. 2011. Sketch-based warping of rgbn images. *Graphical Models 4*, 97–110.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph. 25* (July), 533–540.

SÝKORA, D., DINGLIANA, J., AND COLLINS, S. 2009. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *NPAR '09: Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, 25–33.

THOMAS, B. H., AND CALDER, P. 1995. Animating direct manipulation interfaces. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, 3–12.

VASILAKIS, A., AND FUDOS, I. 2009. Skeleton-based rigid skinning for character animation. In *GRAPP'09*, 302–308.

WANG, Y., XU, K., XIONG, Y., AND CHENG, Z.-Q. 2008. 2d shape deformation based on rigid square matching. *Comput. Animat. Virtual Worlds 19*, 3-4, 411–420.

WENG, Y., XU, W., WU, Y., ZHOU, K., AND GUO, B. 2006. 2d shape deformation using nonlinear least squares optimization. *Vis. Comput. 22*, 9, 653–660.

YAMAUCHI, H., HABER, J., AND SEIDEL, H.-P. 2003. Image restoration using multiresolution texture synthesis and image inpainting. In *Proc. Computer Graphics International*, 120–125.

YAN, H.-B., HU, S., MARTIN, R. R., AND YANG, Y.-L. 2008. Shape deformation using a skeleton to drive simplex transformations. *IEEE Transactions on Visualization and Computer Graphics 14*, 693–706.

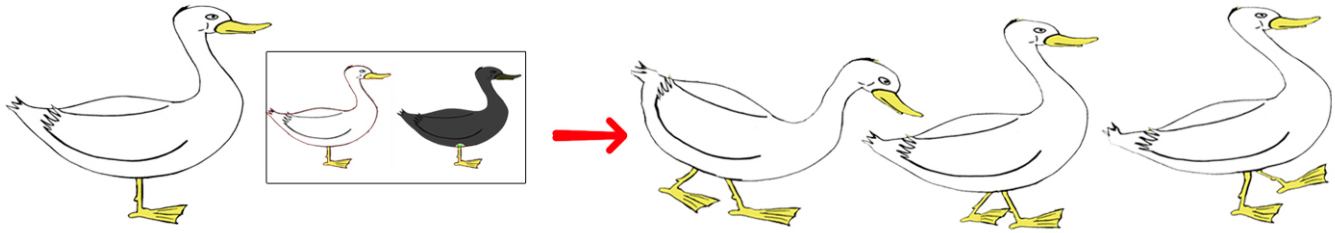


Figure 10: *Example of deformation using two related layers .*

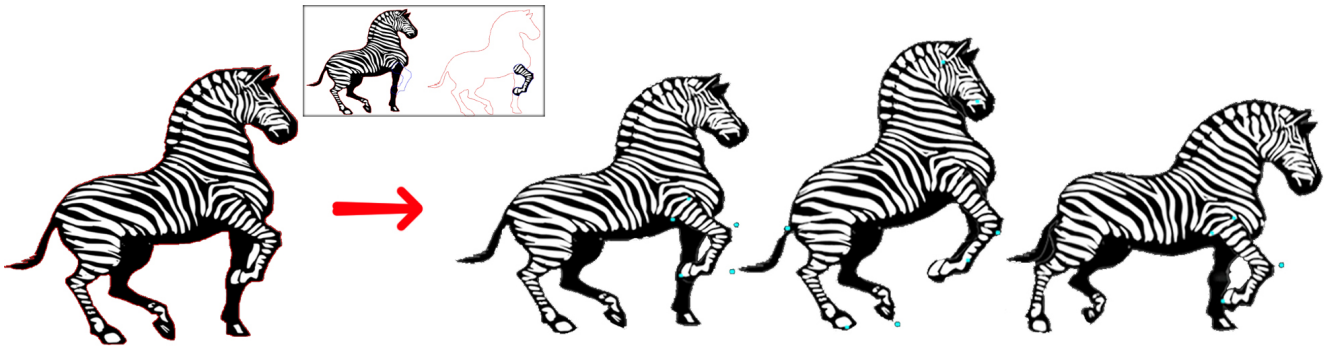


Figure 11: *Example of deformation using two unrelated layers .*

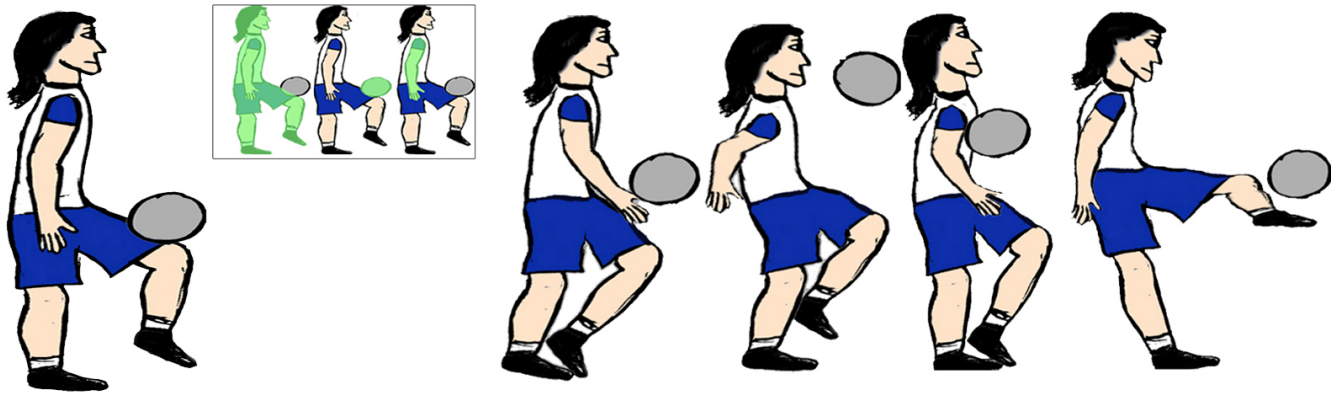


Figure 12: *Example of deformation using three unrelated layers .*

Figure 13: *Example of deformation using a larger numbers of layers (four).*