Interactive As-Rigid-As-Possible Image Deformation and Registration

Marek Dvorožňák*
Supervised by: Daniel Sýkora†

Department of Computer Graphics and Interaction
Faculty of Electrical Engineering
Czech Technical University
Prague / Czech Republic

Abstract

This paper focuses on an existing as-rigid-as-possible deformation model that is particularly suitable for manipulating images that capture articulated objects, for example hand-drawn figures. The model can be used for interactive image deformation as well as automatic image registration. We have implemented both applications as tools for free/open-source image editor GIMP. We describe some details of the implementation and demonstrate functionality of these new tools on a variety of images. For image registration we compare the results of the method with results produced by two existing deformable image registration tools NiftyReg and Drop.

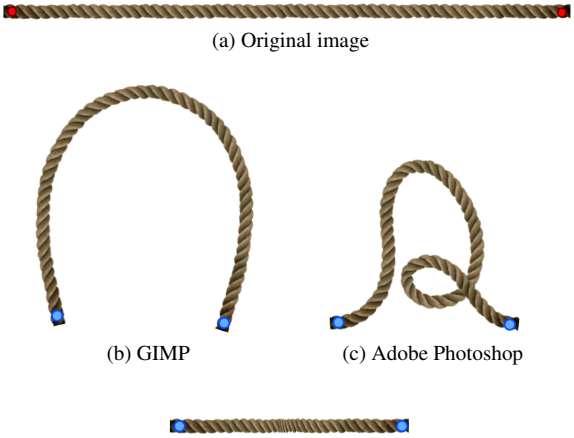
Keywords: image deformation, image registration, as-rigid-as-possible, GIMP

1 Introduction

Image deformation tools implemented in various image editing software allows us to deform image in several ways. Common deformation options include translation, rotation, scaling, shearing and perspective deformation. Recently, various methods have been published which allow user to deform image in a less constrained manner.

In this paper we focus on methods that respect as-rigid-as-possible (ARAP) principle [1]. Its aim is to minimize the amount of local shearing and scaling involved in the deformation. These methods allow user to deform image in a way that during the deformation it behaves like a real world object which is made of rubber.

Some of these ARAP methods are incorporated in recent versions of image editing software. For example, since CS5 version Adobe Photoshop offers Puppet Warp deformation tool based on a method by Igarashi et al. [4], Fiji, a package of tools for image processing, includes Interactive Moving Least Squares deformation plug-in based on Moving Least Squares (MLS) deformation by Schaefer et al. [10] and since 2.7 version graphics editor Krita has Warp tool which employs MLS as well.



(a) Original image

(b) GIMP (c) Adobe Photoshop

(d) Krita

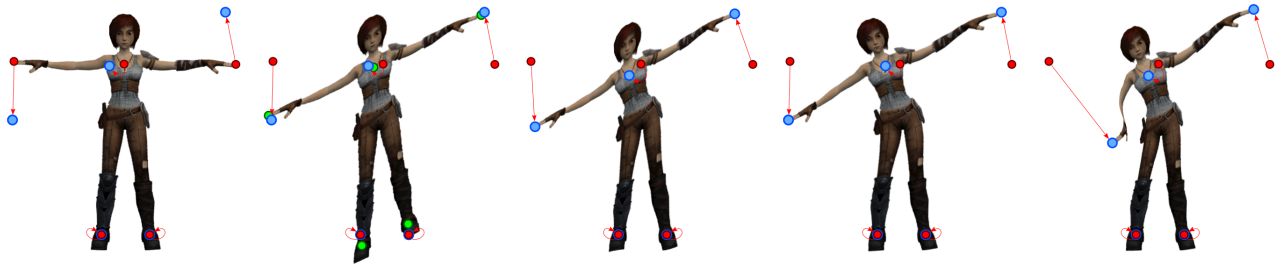
Figure 1: Example of 2-point deformation of a rope using (**b**) our new ARAP deformation tool for GIMP, (**c**) Puppet Warp in Adobe Photoshop and (**d**) Warp tool in Krita.

Free/open-source graphics editor GIMP contains Cage Tool that employs Green Coordinates [5]. The tool allows user to deform image using a cage. The Cage Tool does not preserve as-rigid-as-possible model and thus it is more difficult to obtain realistic deformations of images capturing real world objects. Furthermore, from user point of view, the process of deformation using Cage Tool is relatively cumbersome since at first user has to manually create a cage surrounding the deformed object and only after that he can start deforming the cage using points it is composed of.

As GIMP did not offer an option to deform images in ARAP manner we implemented a tool for ARAP deformation based on a method presented in Wang et al. [13] which allows better deformation results then the aforementioned tools – see Figure 1 for example.

One of the applications of ARAP deformation is ARAP image registration. We extended the ARAP deformation

*dvoromar@fel.cvut.cz
†sykorad@fel.cvut.cz



(a) (b) (c) (d) (e)

Figure 2: Multipoint deformation of image: (**a**) user specifies initial positions of control points (red) and then moves them onto new positions (blue), (**b**) result of a single affine deformation, (**c**) result of affine MLS deformation, (**d**) result of rigid MLS deformation, (**e**) result of large-scale MLS deformation. The image comes from short computer animated film Sintel.

tool and implemented a tool allowing ARAP registration into GIMP. The implementation is based on an image registration method by Sýkora et al. [12].

The paper is organized as follows. First we focus on some of the methods standing behind ARAP deformation tools and also the ARAP image registration method. Then we describe some details of implementation of the deformation and registration tools into GIMP. Finally we demonstrate functionality of these tools on images of various kinds and compare our image registration results with results produced by two existing deformable image registration tools NiftyReg and Drop.

2 Related Work

As-rigid-as-possible (ARAP) deformation principle was introduced by Alexa et al. [1]. Igarashi et al. [4] later employed this principle in deformation of images capturing articulated objects. They use a triangular mesh respecting boundaries of image. User can fix some mesh vertices and move them onto new locations. Then the following operations are performed: (1) similarity transformation is computed for every triangle and (2) the scaling is removed. According to size of triangles, the deformation need not to be smooth. Schaefer et al. [10] employed Moving Least Squares optimization to produce smooth deformations. They solve an optimization problem for every pixel of image using a closed-form formula which they formulated. However, their method cannot handle large-scale deformations. Sorkine and Alexa [11] formulated ARAP deformation as a non-linear optimization problem and presented how to solve it effectively in iterative manner. Wang et al. [13] used square lattice to compute approximation of the non-linear problem for image deformation. They compute rotations for each square on this lattice using shape matching algorithm proposed by Müller et al. [9] by facilitating the closed-form formula for rotation introduced by Schaefer et al. [10].

In image registration, the goal is to find a deformation (and its parameters) of source image (S) that well aligns it with target image (T). Image registration methods often somehow include *deformation model*, image *similarity measure* and *optimization method*.

There are two basic kinds of image registration methods – feature-based and intensity-based [14]. Feature-based methods use features in source and target image. These features have to be detected and the most correct match of features from one image to the other has to be found. For that purpose SIFT keys [6] are frequently employed. Once we have the features and their correspondences, parameters of a mapping function of a selected deformation model can be found employing the Least Squares method or some other method of parameter estimation [3].

Intensity-based methods work directly with intensities of pixels in image. When source and target image differ only in translation (or also slight rotation), it is possible to determine *globally* optimal shift vector by employing simple block-matching method. However, the method has a high time complexity. Another option is to construct an energetic function $E(\mathbf{t}) = d\big(S(\mathbf{p}+\mathbf{t}),\ T(\mathbf{p})\big)$, where $d(S,T)$ is a dissimilarity function representing dissimilarity measure of images S and T. To obtain a *locally* optimal shift vector $\mathbf{t}$, we can for example employ the gradient descent optimization method [7]. In the same way it is possible to find parameters of mapping function of more complex deformation models. Two of recent representatives of intensity-based methods yielding good results in medical imaging are a method by Glocker et al. [2] and a method by Modat et al. [8]. To solve a problem of registration of hand-drawn images, Sýkora et al. [12] proposed an intensity-based ARAP image registration method that utilizes the deformation method by Wang et al. [13].

3 Image Deformation

In this section we focus on point-based image deformation methods employing ARAP principle. In Figure 2a

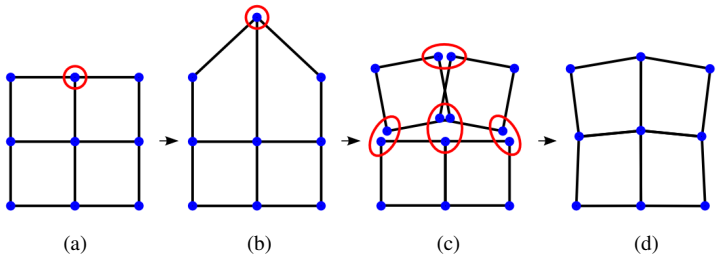


Figure 3: Phases of ARAP deformation of a lattice. Figures (**a**) and (**b**) depict moving one point of the lattice and thus deforming the lattice. Figure (**c**) depicts lattice regularization and (**d**) lattice deformed in as-rigid-as-possible manner. Note that this is just one iteration of the algorithm.

there is an image of a character with control points on it. User first specifies initial positions of these points (indicated by red points) and then he moves these points onto new positions (indicated by blue points) and thus expects to deform the image. Our ultimate goal is to deform the image so that these user specified constraints are satisfied and the amount of local shearing and scaling involved in the deformation is minimized.

3.1 Moving Least Squares Deformation

Our task is to find such affine transformation of coordinates (i.e. transformation matrix $\mathbf{A}$ and translation vector $\mathbf{t}$) which moves red points $\mathbf{p}_i$ so that they are located as close as possible to blue points $\mathbf{q}_i$. Schaefer et al. [10] employed the (Moving) Least Squares method and algebraically formulated the problem as

$$\arg\min_{\mathbf{A},\mathbf{t}} \sum_i w_i \left(\mathbf{A}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\right)^2. \tag{1}$$

The weight w_i is defined as

$$w_i = \frac{1}{(\mathbf{p}_i - \mathbf{v})^{2\alpha}}. \tag{2}$$

We can see that it is a function which yields very high values near points $\mathbf{p}_i$. The nearer the pixel $\mathbf{v}$ to some point $\mathbf{p}_j$, the higher is the influence of an affine transformation that maps point $\mathbf{p}_j$ to point $\mathbf{q}_j$; α is a selected parameter.

To simplify the solution of the minimization problem, we first solve for $\mathbf{t}$ and obtain optimal vector of translation $\mathbf{t}_{\min} = \mathbf{q}_c - \mathbf{A}\mathbf{p}_c$, where $\mathbf{q}_c$ and $\mathbf{p}_c$ are weighted centroids of positions of control points, i.e. $\mathbf{p}_c = \frac{\sum_i w_i \mathbf{p}_i}{\sum_i w_i}$, $\mathbf{q}_c = \frac{\sum_i w_i \mathbf{q}_i}{\sum_i w_i}$. After that we can remove the translation from the equation which yields $\arg\min_{\mathbf{A},\mathbf{t}} \sum_i w_i \left(\mathbf{A}\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_i\right)^2$, where $\hat{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}_c$, $\hat{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{q}_c$. Then we can solve for $\mathbf{A}$ and obtain optimal transformation matrix $\mathbf{A}_{\min} = \sum_i \left(w_i \hat{\mathbf{q}}_i \hat{\mathbf{p}}_i^T\right) \cdot \left(\sum_i w_i \hat{\mathbf{p}}_i \hat{\mathbf{p}}_i^T\right)^{-1}$.

Figure 2b depicts a result of applying this method on our image with $w_i = 1$. That corresponds to a deformation produced by a single affine transformation. Figure 2c shows a result of the deformation with weights defined as in (2). The result contains an undesirable shearing visible in it.

To obtain a better looking result, it is necessary to extract a rigid transformation out of the affine transformation. For that purpose, matrix decomposition methods such as singular value decomposition or polar decomposition can be employed.

In 2D, closed-form formula that can be used to obtain the rigid transformation directly was presented by Schaefer et al. [10]. Transformation matrix of the rigid transformation is formulated as

$$\mathbf{R}_{\min} = \frac{1}{\mu} \sum_i w_i \begin{pmatrix} \hat{\mathbf{p}}_i \\ -\hat{\mathbf{p}}_i^{\perp} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{q}}_i^T & -\hat{\mathbf{q}}_i^{\perp T} \end{pmatrix} \tag{3}$$

where

$$\mu = \sqrt{\left(\sum_i w_i \mathbf{q}_i \hat{\mathbf{p}}_i^T\right)^2 + \left(\sum_i w_i \mathbf{q}_i \hat{\mathbf{p}}_i^{\perp T}\right)^2} \tag{4}$$

and operator $\perp$ represents a perpendicular vector, i.e. $(x,y)^{\perp} = (y,-x)$.

Figure 2d depicts a result of deformation after processing the image using this as-rigid-as-possible MLS deformation. The result looks more natural.

However, with this approach we only obtain plausible results for suitable positions of points $\mathbf{p}_i$ and $\mathbf{q}_i$. Figure 1d depicts a case where MLS deformation cannot yield desired result because of non-linearity of the problem [11].

Another problem that appears here is caused by the measure (Euclidean distance) that is employed in the weighting function (2). This measure does not respect a topology of image. The consequence of this is that when we e.g. deform an image of a character and we move a point located on a hand towards the body, not only the hand is deformed but also hips and the body (see Figure 2e). This can be solved by a measure that respects image topology.

3.2 Rigid Square Matching

A method that does not suffer from the aforementioned problems was presented by Wang et al. [13]. Examples of

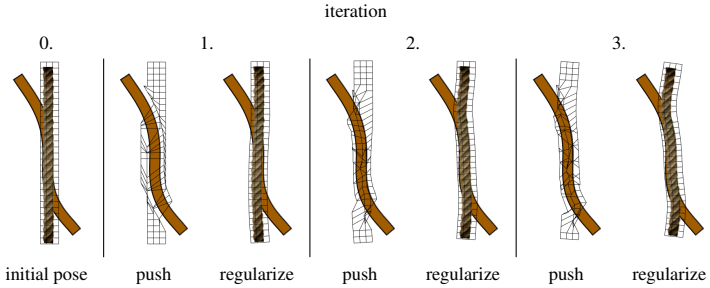


Figure 4: Several first iterations of ARAP image registration algorithm. The goal is to deform the image of a rope to well align it with the image of bent rectangle.

image deformations produced by this method are depicted in Figure 5.

In this method a square lattice (mesh) copying shape of image is firstly constructed above image. Some points of the lattice are moved by user to desired locations and the lattice is then iteratively deformed in as-rigid-as-possible manner. Deformed image is obtained by using any of suitable 4-point image deformation methods on each square of the lattice.

The mesh composed of squares is used despite the fact that with it we are not able to precisely copy the image (as with triangular mesh). An advantage of square mesh is simplicity of its construction (in comparison to e.g. triangulation) and a lower number of lattice elements in comparison to triangular mesh. A shape-aware mesh splitting algorithm [13] can be employed to achieve similar behaviour as with triangular mesh.

A lattice is actually a group of points. Similarly to section 3.1 we have initial positions $\mathbf{p}_i$ of points of the lattice (or alternatively *source* lattice) and target positions $\mathbf{q}_i$ of the lattice (or *target* lattice). User sets new positions of some of the points $\mathbf{q}_i$ of the target lattice (Figure 3a) and thus deforms the lattice – some of the lattice squares become quadrilaterals (see Figure 3b).

The core of the method is in a regularization phase where we try to respect user specified constraints in a form of the placed points $\mathbf{q}_i$ as well as deform the lattice in a way that every single square (quadrilateral) is deformed the most rigidly. We perform a specified number of regularizing iterations of which every one performs the following operations:

1. Rigid transformation of every lattice square (see Figure 3c) using the formula (3) and $w_i = 1$. This transforms every quadrilateral to square again.

2. Centering every originally overlapping points of the lattice (see Figure 3d). This ensures that points in every cluster of originally overlapping points of the lattice will overlap again – and squares become quadrilaterals again.

Hundreds of these iterations are usually performed. Convergence of the method is slower than with ARAP deformation methods that solve linear system [11] instead of computing centroids. However, the advantage of the method is that it does not require us to specify fixed points. This fact is utilized in image registration [12].

4 ARAP Image Registration

In this section we will focus on an application of ARAP image deformation. We will describe an intensity-based registration method presented by Sýkora et al. [12]. The method was invented for usage in cartoon industry and is especially suitable for registration of hand-drawn articulated images.

When registering such images, it is not possible to use methods based on features since every drawing is unique to some extent and hence it is not possible to find corresponding features. In such a case methods based on optimization of energetic function may lead to a success. However, if source and target image differ in large non-linear deformation, these methods usually get stuck in a local extrema and hence the result of these methods will often not be plausible – see Figure 7f.

The method by Sýkora et al. employs the rigid square matching deformation algorithm described in section 3.2, sum of absolute differences (SAD) dissimilarity measure and block-matching optimization method. The method takes advantage of the fact that the deformation algorithm allows us to arbitrarily move points $\mathbf{q}_i$ without having to consider their mutual connection. The actual registration is divided into "push" and "regularize" phases. These two phases are continuously performed until fulfillment of a stop condition.

In Figure 4 there are two overlapping objects depicted – image of a rope (source image) and image of a bent rectangle (target image). The aim of this registration problem is to deform the image of a rope so that it well aligns with the image of a bent rectangle.

In the *push* phase of ARAP registration, we move lattice points $\mathbf{q}_i$ to locations where the area of the source image around these points differs *as little as possible* from the area of the target image (around these points). To find such a translation vector $\mathbf{t}$ the method employs the block-matching method that finds the optimal translation

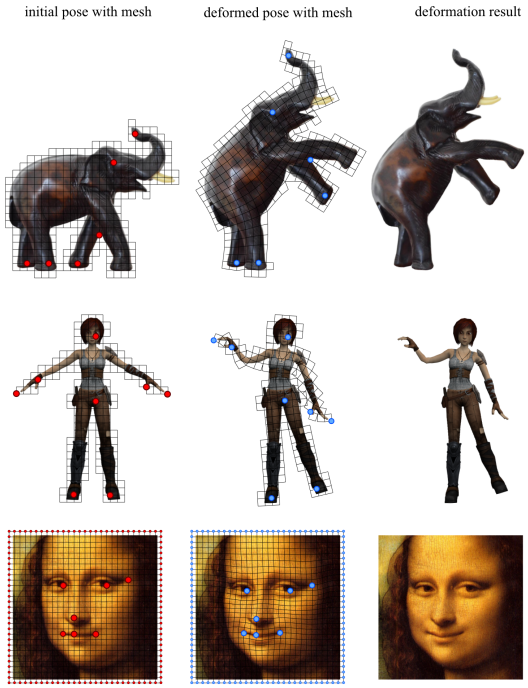


Figure 5: Examples of image deformation created using our N-Point Deformation tool in GIMP. Red points indicate initial positions of points that are *fixed*, blue points indicate desired positions of these points.

vector in defined search area. For SAD dissimilarity measure the optimal translation vector $\mathbf{t}_{\text{opt}}$ can be formulated as

$$\mathbf{t}_{\text{opt}} = \arg\min_{\mathbf{t}\in M} \sum_{\mathbf{p}\in N} |S(\mathbf{p}+\mathbf{t}) - T(\mathbf{p})|, \qquad (5)$$

where M is a search area where we search for optimal shift, N is SAD "neighborhood", S is source and T is target image. In *regularization* phase the lattice is put into a consistent state by the rigid square matching algorithm. Figure 4 shows several first iterations of ARAP image registration algorithm together with results of push and regularize phases for each iteration.

By the number regularizing iterations in the rigid square matching algorithm, we can adjust rigidity (or flexibility) of deformation. That is utilized in this image registration method to refine the result – see Figure 7, 8, 9, 10.

5 Implementation

Our goal was to implement the rigid square matching algorithm and the ARAP image registration algorithm into GIMP. For implementation C programming language was selected since GIMP is written in this language.

We implemented (1) the rigid square matching algorithm into a new library named *libnpd*, (2) an operation that allows ARAP image deformation into GEGL library which is employed in GIMP; the operation utilizes libnpd library, (3) ARAP image deformation tool into GIMP; the tool utilizies the aforementioned operation, (4) ARAP image registration tool into GIMP; the tool extends the deformation tool.

In this paper, we use the term "n-point deformation" for image deformation employing the rigid square matching algorithm and the term "n-point registration" for image registration employing the ARAP image registration algorithm.

5.1 N-Point Deformation Library (libnpd)

The library contains data types and functions allowing to perform n-point image deformation and registration. The library is designed in a way so that it can be used with various graphics libraries and thus requires to implement some graphics functions (e.g. get_pixel_color, set_pixel_color) and data types (image, display).

One of the most important data types in the library is NPDModel type which holds source lattice, target lattice (i.e. "source" and "target" group of points), source image and display.

5.2 GIMP and GEGL Library

As already stated, GIMP is written in C (C89 standard). The C language itself is not object-oriented. GIMP uses the C language enriched with object-oriented approach using GObject object system, which is part of GLib library.

GIMP currently uses GEGL and BABL libraries which should allow it to work with high depth color images and use some non-destructive image editing techniques. GEGL is a graphics library which uses specified oriented acyclic graph to process images. This graph is made up of individual nodes that can represent graphics operations as well as another graph. Edges that connect individual nodes determine the order in which the graph will be processed. Nodes usually expect image on their inputs and produce (filtered) image on their output.

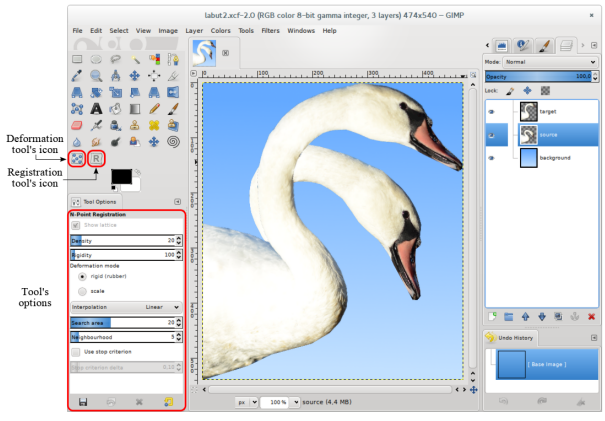


Figure 6: Our ARAP image deformation and registration tools integrated into the environment of GIMP

(a) source image (b) target image (c) initial overlap (d) NPR_1 (e) NPR_2 (f) Drop (g) NiftyReg

Figure 7: Registration of a hand-drawn images. The task is to deform the source image to well align it with the target image. Resolution of images: 420×541. NPR_1 is a result of NPR tool with rigidity set to 600, for NPR_2 the rigidity is 30. Courtesy of Universal Production Partners.

(a) source image (b) target image (c) initial overlap (d) NPR_1 (e) NPR_2 (f) Drop (g) NiftyReg

Figure 8: Registration of a hand-drawn images. Resolution of images: 560×400. NPR_1 is a result of NPR tool with rigidity set to 200, for NPR_2 the rigidity is 30. Courtesy of AniFilm.

Functionality of GIMP can be extended using plug-ins. They can be written in various programming languages – namely Scheme, Python, Perl and C. GIMP incorporates a large amount of plug-ins, most of them function as graphic filters. The current development trend is not to create graphic filters in form of GIMP plug-ins but instead to create GEGL operations within GEGL library.

N-point deformation algorithm was implemented into the GEGL library as a new operation. This operation employs libnpd library. There can be two scenarios of its usage depending on whether we *have* or we *do not have* NPDModel.

When we do not have NPDModel, the operation expects image on its input. After the first processing of a graph the operation is contained in, the operation creates NPDModel and returns it through operation's parameters. User can then deform the target lattice contained in NPDModel by manipulating with lattice's points. Second and another processing produces ARAP deformed image that is result of a specified number of deformation iterations. This image is available on operation's output.

When we have NPDModel, we can supply it to the operation through operation's parameter and perform the deformation in the same way as described in previous paragraph.

5.3 N-Point Deformation Tool

N-point deformation was implemented as internal tool (instead of a plug-in). That allows the tool's GUI to be seamlessly integrated in GIMP. Individual control points can be placed directly on the canvas. During the deformation process, user can use GIMP's GUI e.g. to easily zoom to a certain part of image that is being deformed and focus on details or he can arbitrarily rotate the canvas.

Every tool in GIMP is implemented as a class extending a parent class named GimpTool. This parent class provides the basic functionality common to all tools in GIMP. This class is extended by a class named GimpDrawTool that allows its descendant classes to add GUI elements on canvas and that is able to draw these elements. These elements include control points (handles), basic plane shapes, guide lines, paths, text cursor and also a live preview of a result of operation that is performed by a tool. This class is extended by various classes representing tools, let us mention for example a group of painting tools, selection tools, transformation tools and also a tool for writing text.

The class GimpDrawTool is also extended by our class named GimpNPointDeformationTool that implements the N-Point Deformation tool. This class employs libnpd library. Using a deformation thread, it performs a deformation of image, using a preview thread, it draws at regular intervals a preview of current state of the deformation. The preview thread calls the methods of GimpDrawTool in order to redraw GIMP's GUI.

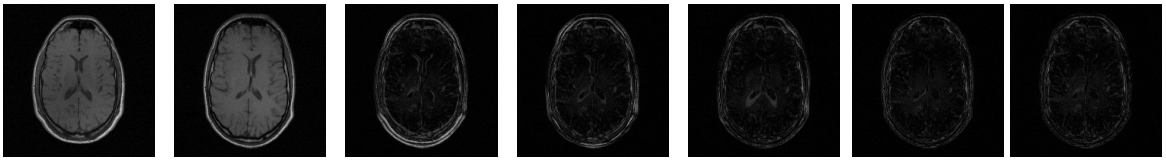
Every GIMP tool can define its own set of settings and their graphic representation within GIMP tool's GUI. For these purposes, a class named GimpToolOptions is employed. This class is inherited by classes describing individual tools options. N-Point Deformation tool employs its own class named GimpNPointDeformationOptions.

5.4 N-Point Registration Tool

As the *n*-point deformation, the *n*-point registration was implemented into GIMP as an internal tool. The main rea-

(a) source image (b) target image (c) initial overlap (d) NPR_1 (e) NPR_2 (f) Drop (g) NiftyReg

Figure 9: Registration of images of a person. Resolution of images: 489×656. NPR_1 is a result of NPR tool with rigidity set to 200, for NPR_2 the rigidity is 30.



(a) source image (b) target image (c) initial overlap (d) NPR_1 (e) NPR_2 (f) Drop (g) NiftyReg

Figure 10: Registration of images of a brain. Resolution of images: 421×436. NPR_1 is a result of NPR tool with rigidity set to 50, for NPR_2 the rigidity is 5. Difference images have been used to show the resulting overlap. Images come from the RIRE dataset.

sons for this decision included the possibility to use existing implementation of N-Point Deformation tool and the possibility to allow user to easily help the registration (by moving one or several points to correct locations) in situation when the registration get stuck in undesirable state.

A class named GimpNPointRegistrationTool which extends GimpNPointDeformationTool class was created. The N-Point Deformation tool class was modified to allow its employment for registration purposes. As previously mentioned, the N-Point Deformation tool uses a thread to perform the deformation. Within this thread a method performing the deformation is being called. This method is overridden in GimpNPointRegistrationTool by a method performing the ARAP image registration algorithm.

A class GimpNPointRegistrationOptions defining a set of settings of the tool was created. This class extends GimpNPointDeformationOptions class which was modified to allow its subclasses to use a set of generic settings of the deformation.

6 Results

We used our N-Point Deformation tool to deform several images – see Figure 1 and 5. Let us look at Figure 1 depicting a deformation of a rope using various deformation tools. Krita (and Fiji) produced undesirable result. With Adobe Photoshop we can obtain a result similar to ours, however, the process is cumbersome. With Puppet Warp tool user must rotate control points (pins) to achieve desired deformation. There is an automatic pin rotating function available, however, as in our example it might not work correctly. There is no need to rotate control points in N-Point Deformation tool. Thanks to the method employed in our tool, the deformation is predictable and the deformation process is easier than with Puppet Warp in Adobe Photoshop.

We compared our N-Point Registration (NPR) tool with deformable image registration tools Drop [2] and NiftyReg [8]. Figure 7 shows a plausible result of NPR tool. Even we tried to set the best parameters in Drop and NiftyReg tools we obtained unsatisfactory results. Here, the deformation model preserving rigidity is a great advantage. Figure 8 shows an example where all three tools gave satisfactory result. There we can see that Drop and NiftyReg produced results that look more similar to the target image. However, the details are more distorted. Figure 9 shows registration of images of a person. In this problem we obtained plausible results with NPR and Drop. The latter produced really good result, however, there is again problem with details (left hand). With NiftyReg we could not obtain satisfactory result again. In a registration problem involving registration of articulated images, good approach is to first perform ARAP registration and then refine with a non-linear registration method as in Drop or NiftyReg. Sýkora et al. use that approach in [12].

Figure 10 shows results of a common registration problem – aligning two medical images. Drop and NiftyReg were designed exactly for this kind of a problem and gave very good results. We can see that NPR can also handle this kind of registration problem because the rigid square matching algorithm allows some amount of shrinking or stretching (depending on a number of deformation iterations). However, the result is not as good as with the two other tools.

7 Conclusions and Future Work

We implemented as-rigid-as-possible image deformation and registration tools into a development version of popular free/open-source image editor GIMP and demonstrated their functionality on images of various kinds.

For image registration it is evident from the results that some modern non-linear image registration methods, when properly set, are able to cope even with a large deformation of images entering the registration. In contrast to these methods in some situations a great advantage of the ARAP image registration method can be the fact that in almost all circumstances it produces results that are not unnaturally deformed. This is particularly useful when registering real or cartoon figures and their poses.

Although the tools are functional they still have some weaknesses. Further work is thus to eliminate them. The tools have currently problems regarding speed when working with large images, mainly due to redrawing the preview of the deformation. The preview has to be rendered several times per second, which is what causes the problems. In the registration tool the problem is also caused by block-matching method that is employed during registration. For large images, it is necessary to set a higher value of the search parameters ("search area" and "neighborhood"). Currently, user does not have an option to set a depth of individual control points and thus he cannot specify which part of the overlapping lattice (image) will be visible. The tools currently do not use multiple CPU/GPU cores to speed up their computations.

Acknowledgements

Part of the work on N-Point Deformation tool has been supported by Google through Google Summer of Code 2013. Thank must go to GIMP developers for selecting the project.

This work has been partially supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS13/214/OHK3/3T/13 (Research of Progressive Computer Graphics Methods).

References

[1] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *ACM SIGGRAPH Conference Proceedings*, pages 157–164, 2000.

[2] Ben Glocker, Nikos Komodakis, Georgios Tziritas, Nassir Navab, and Nikos Paragios. Dense image registration through MRFs and efficient linear programming. *Medical Image Analysis*, 12(6):731–741, 2008.

[3] A. Ardeshir Goshtasby. Robust parameter estimation. In *Image Registration*, Advances in Computer Vision and Pattern Recognition, pages 313–341. 2012.

[4] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, 2005.

[5] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Transactions on Graphics*, 27(3):78, 2008.

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[7] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, volume 2, pages 674–679, 1981.

[8] Marc Modat, Gerard R. Ridgway, Zeike A. Taylor, Manja Lehmann, Josephine Barnes, David J. Hawkes, Nick C. Fox, and Sébastien Ourselin. Fast free-form deformation using graphics processing units. *Computer Methods and Programs in Biomedicine*, 98(3):278–284, 2010.

[9] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics*, 24(3):471–478, 2005.

[10] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Transactions on Graphics*, 25(3):533–540, 2006.

[11] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, pages 109–116, 2007.

[12] Daniel Sýkora, John Dingliana, and Steven Collins. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, pages 25–33, 2009.

[13] Yanzhen Wang, Kai Xu, Yueshan Xiong, and Zhi-Quan Cheng. 2D shape deformation based on rigid square matching. *Computer Animation and Virtual Worlds*, 19(3–4):411–420, 2008.

[14] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.