A new measure for distance-field based shape matching

Pavel Voronin
Kurchatov NBICS Center
pavel.voronin@gmail.com

Andrew Adinetz
Research Computing Center,
Lomonosov Moscow State University
Joint Institute for Nuclear Research
adinetz@gmail.com

Dmitry P. Vetrov
Lomonosov Moscow State University
Kurchatov NBICS Center
vetrovd@yandex.ru

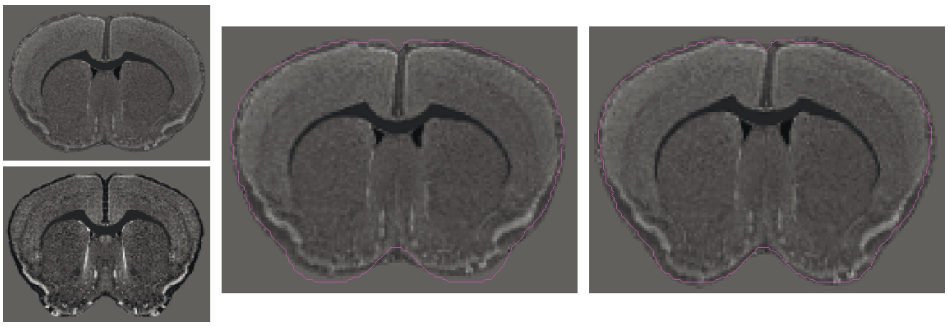

Figure 1: *Shape registration via distance field matching: top left, source image; bottom left, target image; middle, source registered onto target, using the classic dissimilarity measure; right, source registered onto target, using the new dissimilarity measure. The target's contour is superimposed in magenta.*

Abstract

One popular method to match two shapes is to register their distance fields as images. We discuss the well-known robustness problems of this approach and identify the noncommutativity of distance transform and geometric transformations as a core issue. Building on this, we propose a simple modification of the method, deriving a new dissimilarity measure. As it involves multiple distance field computations, we also present an efficient GPU-based algorithm for this problem.

Keywords: shape matching, registration, distance fields

1 Introduction

Shape registration (or *shape matching*) is one of the fundamental computer vision problems: given two figures, transform one of them, *source*, into a figure as similar in shape to the other one, *target*, as possible. This task arises naturally in such fields as shape reconstruction [Levoy 2000], shape tracking [Zhou 2005], shape interpolation [Kilian 2007], semantic deformation transfer [Baran 2009], shape recognition [Fahmi 2008], shape retrieval [Belongie 2001], and statistical shape modeling [Zhu 1996]. Combined with intensity-based methods, it has also been widely used for image segmentation [Tsai 2003] and registration [Babalola 2006]. Some of the most important applications of these methods are in medical imaging and related areas [Maintz 1998, Zitova 2003], including data alignment [Malcolm 2008], detection of tumors and nodules [Ginneken 2001], and guidance in neurosurgery [St-Jean 1998]. Our particular interest lies in automatic or user-assisted mapping of experimental brain images onto an atlas (Fig. 1).

More formally, the problem of shape registration can be reformulated as follows. For a pair of admissible figures $A, B \in \Upsilon$, and a permissible set of transformations Ω, find a transformation $T^{opt} \in \Omega$ that, when applied to A, results in a shape that minimizes a certain dissimilarity measure F_{dis}, regularized by a smoothness measure F_{sm}, with respect to B:

$$T^{opt} = \underset{T \subset \Omega}{argmin} \left(F_{dis}\left(T\left(A\right), B\right) + \lambda \cdot F_{sm}\left(T\right)\right). \quad (1)$$

We will use integral of the sum of squared second derivatives as a smoothness measure, which is a popular, but by no means the only, choise [Munim 2007, Paragios 2003].

Design and complexity of shape registration are largely defined by the permissible set of transformations, or *transformation model* [Zitova 2003]. It is usually determined by the data acquisition process and shape variability among the objects being registered. Linear tranformations include rigid motions, similarity transforms, affine transforms and perspective projections; these are global mappings defined by a small number of parameters (3, 4, 6, and 8, accordingly, in the planar case). Non-linear transformations can be explicitly parameterized, usually by tens to hundreds variables, each influencing a small region (examples include polynomial splines, radial basis functions and partition-of-unity). Another option is to derive the transformation from a physical model of the objects being registered, by defining external stretching forces, working towards improving local similarity, and internal resisting forces, minimizing amount of bending and stretching — and iterating towards the minimum energy state (popular examples of the models include those based on elastic rubber sheet and viscous fluid).

Shape representation is another important factor. Point clouds, spectral descriptors, parametrical curves, snake models, shock graphs and skeletons have all been extensively used in shape matching [Veltkamp 1999]. A more recent approach is to represent the figure's border implicitly as zero level-set of its distance field [Paragios 2003]. Distance field of a closed planar contour γ is a scalar function defined for any point x as follows:

$$DF\left(x;\gamma\right) = \sigma\left(x;\gamma\right) \cdot dist\left(x,\gamma\right); \quad (2)$$

$$\text{for } x \text{ inside } \gamma,\ \sigma\left(x;\gamma\right) = -1;\ \text{otherwise, } \sigma\left(x;\gamma\right) = 1.$$

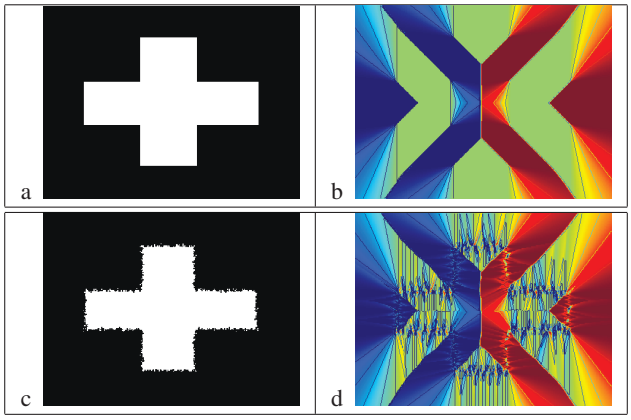

Figure 2: *Distant fields are not robust to noise: left, source figure and its noisy counterpart; right, first components of the gradients of their distant fields.*

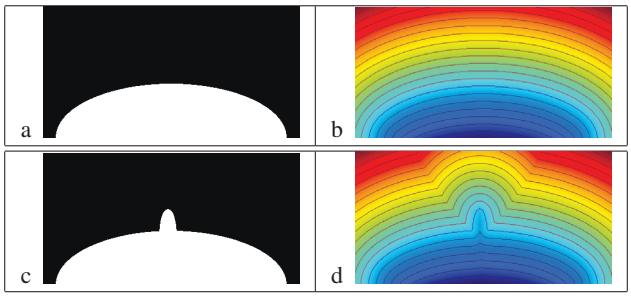

Figure 3: *A localized change in shape (a, c) can have a far-reaching effect on its distant field (b, d).*

The main advantage of using distance fields for shape registration is that by discretizing the field on a rectangular support we can recast the problem as one of image registration, and draw upon the vast literature on the subject. Dissimilarity measure for such *distance images*[1] is usually the sum-of-squared-distances (SSD), or, in case of siginificant variation in shapes' scale, mutual information [Huang 2006] (correlation ratio was also discussed, but not implemented, in [Hong 2006]). Optimization methods for both linear and various nonlinear transformations have been proposed [Paragios 2003, Huang 2006, Liu 2011].

2 Analysis

Combining equations 1 and 2, we get the following expression for the transformation matching contours A and B:

$$T^{opt} = \underset{T \subset \Omega}{argmin} \left(F_{dis} \left(T \left(DF \left(A \right) \right), DF \left(B \right) \right) + F_{sm} \left(T \right) \right). \tag{3}$$

This approach presents certain problems. First of all, distance fields are not robust to small noise-like perturbations [Hong 2006]. Figure 2 demonstrates this effect: given a simple figure (a), that defines a distance field with smooth gradients (b), we add only a modest amount of binary noise to the figure's contour (c), which results in significant degradation of the gradients (d). Effects of the noise

[1]We'll use $DF(\gamma)$ to refer to the distance image of γ. The image's dimensions are defined by the application.

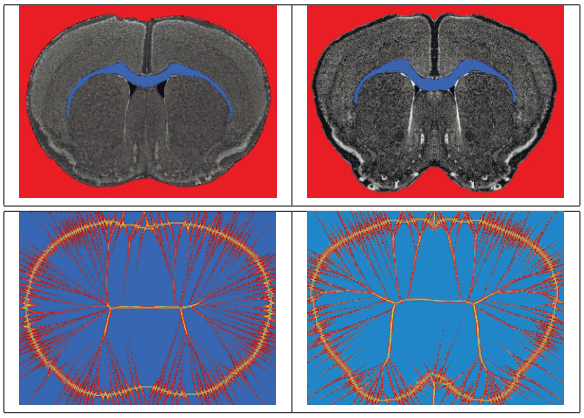

Figure 4: *Two similarly shaped figures show pronounced differences in the structure of their distance fields (shown: magnitudes of gradients, contrast-enhanced).*

are not local to the contour, effectively changing the distance field's structure.

What's more, you don't need to add distortions all over the contour to affect its distance field [Liu 2011]. As figure 3 illustrates, a small localized tranformation of a contour can change values and smoothness of a sizable part of its distance field.

A more general fact, related the previous two, is as follows: two figures can have similar shape, but differently structured distance fields. In figure 4, this is demonstrated by comparing a pair of brain slices. Taken from two different brains, but in anatomically close positions, their shape is roughly equivalent. At the same time, their distance fields have markedly distinct features. No geometrical transformation can match these fields exactly. For shape registration, this can result in slower optimization and suboptimal fit.

Limitations of the approach can be summed up by a simple observation [Liu 2011]. For any transformation T acting upon contour A in a non-rigid way:

$$T \left(DF \left(A \right) \right) \neq DF \left(T \left(A \right) \right). \tag{4}$$

In other words, transformation of a shape's distance field is no more than an estimate of the distance field of the actual transformed shape. As a consequence, dissimilarity measure in equation 3 only approximates the actual dissimilarity, imposed by T.

At least three basic approaches have been employed by the research community to improve the method's robustness: limiting the calculation of the dissimilarity measure to narrow bands around the contours [Paragios 2003]; changing the shape representation to a generalized version of distance fields (integral kernels [Hong 2006], vector distance functions [Munim 2007]); changing the dissimilarity measure to a more robust one (mutual information [Huang 2006], variational chamfer-matching energy [Liu 2011]). Of these, only the latter work addresses property 4, albeit somewhat indirectly. Authors propose a symmetric variational measure, generalizing the "narrow bands" idea (eliminating the need for a heuristic to control the width of the bands). While their method does achieve very promissing results, it uses complicated techniques for shape representation and optimization, making it much more difficult to implement than any other algorithm discussed here.

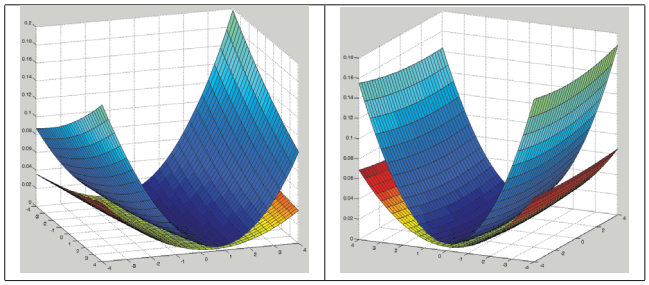

Figure 5: *Comparing dissimilarity measure used in equation 3 (warm colours) to that of equation 5 (colder palette).*

3 Proposed method

Considering the importance of property 4, discussed above, we would like to propose the following simple modification of procedure 3:

$$T^{opt} = \underset{T \subset \Omega}{argmin} \left(F_{dis} \left(DF\left(T\left(A\right)\right), DF\left(B\right)\right) + F_{sm}\left(T\right)\right). \tag{5}$$

While both alternative shape representations and dissimilarity measures, discussed above, could be incorporated into this algorithm, for space and clarity considerations, we shall only discuss the case of distance fields and SSD. When calculating the smoothness measure, we found it useful not to integrate over the whole image, but to exclude a narrow band around the contour. This usually results in better fit and improves convergence.

Although one could match distant field images using any image registration algorithm, most methods discussed above boost perfomance rate by introducing some kind of efficient approximation of the gradient of the dissimilarity measure, often allowing analytical derivation. In our case, though, transformation is incorporated into the measure in a highly non-linear and opaque fashion, which seems to preclude this kind of approach. On the other hand, derivative-free optimization methods remain either unreliable or difficult to tune for expensive functions in higher dimensions. So, we have to fall back to the use of finite differences to approximate the gradient.

Since every component of the gradient needs two evaluations of the dissimilarity measure, we need a tranformation model that has as little parameters as possible, while still being relatively flexible. Ideally, each parameter should also influence only a small part of the image, allowing to group them in batches for simultaneous processing. Classical apparatus of cubic B-splines happens to be just such a model [Huang 2006]. Actually, it has the following neat property: gradient's computational complexity is virtually independent on the number of control points [Knott 2000]. Still, it takes 30+ function evaluations per 1 gradient evaluation. Considering the cost of the gradient computation, as well as the significant dimensionality of the search-space, we chose the limited-memory version of the Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) as the optimizer, since it's particulary suited to this kind of problems [Byrd 1995]. For both L-BFGS optimization and B-spline transformations, we used the efficient implementations published by Dirk-Jan Kroon to accompany his PhD thesis [Kroon 2011].

Unlike in previous approaches, when calculating the distance field for a contour on a grid, we can not assume that the contour's points all come from the grid. Nor can we just project the points onto the grid before the calculation, as we need our measure to be sensitive to small perturbations of the contour (it is crucial for the gradient approximation). Consequently, we can't use the fast image-based distance transform algorithms [Ciesielski 2011], and have to calculate distance field of a polygon instead, which is a very computationally-intensive procedure. In fact, we found that with a straightforward implementation, it takes over 95% of the calculation time. We address this problem by introducing an efficient GPU-based library for computing distance fields, based on NUDA (see section 4). In this, we follow [Voronin 2009], but use more advanced and up-to-date programming libraries and techniques. Note that we shall only discuss the magnitude of the distance field and not the sign (σ in equation 2), since its computation proceeds in a similar way and takes only a small fraction of the combined time.

To compare the two dissimilarity measures, classic $F_{old} = F_{dis}\left(T\left(DF(A), DF\left(B\right)\right)\right)$ of equation 3 and proposed $F_{new} = F_{dis}\left(DF\left(T\left(A\right)\right), DF\left(B\right)\right)$ of equation 5, we conducted the following experiment. Using a sample mouse brain contour, 16x16 B-spline control point grid (512 variables), and 200x200 distance field discretization, and registering the contour upon a transformed version of itself, we studied the measures in vicinity of identity transform when all but two variables (pertaining to a single control point), are fixed. For each control point P_i, we went through 32x32 regularly spaced pairs of values for variables v_i^1 and v_i^2, and calculated F_{new} and F_{old} for each of them. Choosing from the cases that exibited significant variation, in figure 5 we show typical examples as surfaces (while these are all free of local extrema and even convex, this can not be expected in general case). Notably, $F_{new} > F_{old}$ and $\|\nabla F_{new}\| > \|\nabla F_{old}\|$ in all cases. That F_{new} shows steeper behaviour indicates that it is indeed a more sensitive measure of dissimilarity, and is also important for the convergence speed of the optimization procedure.

Using both measures, we've implemented a tool for brain slice alignment. You can see an example of its work in figure 1. While the difference between the results may not always be as pronounced as in this case, in our experience, the new measure consistently outperforms the old one and is more robust. With GPU-optimized distance field computations, coregistering 200x300 slices takes, depending on how different they are, 2-10 seconds.

4 GPU implementation

Nemerle is an extensible language, that is, a language which allows extending its syntax and semantics relatively easily. Nemerle accomplishes this with the help of macros, special functions which execute at compile-time as compiler plugins, perform code transformations and extend syntax of the language. Nemerle also provides code quoting facilities for concise creation of code trees from templates, and concise analysis of code trees passed as macro parameters. Nemerle macros are actually quite rich: any .NET method, either standard or user-written, can be called from inside a macro. Also, many features considered "built-in" in other languages, such as loops, conditionals, locks or asynchronous execution, are in fact implemented as macros in Nemerle.

NUDA (= Nemerle Unified Device Architecture) [Adinetz 2012, Adinetz 2011] is a set of Nemerle extensions for programming graphics processors. Extensions provided include multi-dimensional for-like loops, sending loops to execute on GPU, used-defined on-GPU functions and a set of loop transformations. Simple and structural types, as well as arrays, are supported for use in GPU kernels. libgpuvm enables using of ordinary .NET arrays on GPU with little copying overhead through lazy synchronization and moving data to host with userspace pagefault handling. Special array types automatically synchronized between host and GPU are also provided, though this is currently not a preferred way of using

Algorithm 1 Array addition in NUDA (for better readability, reserved words are in bold type, while the comments are italicized).

```
// allocate arrays
def a = array(n) : array[float];
def b = array(n) : array[float];
def c = array(n) : array[float];
// ... initialization ...
// array addition on GPU
nuwork(64) do(i in n)
c[i] = a[i] + b[i];
// some work with array c
```

Algorithm 2 Different versions of the distance field computation, top to bottom: baseline; optimized for NVidia Fermi; optimized for AMD (optimization-related annotations are undescored for better redability).

```
nuwork(128) do(i in m) {
  def p = ps[i];
  mutable d2 = 1e38f;
  do(j in n1)
    d2 = min(d2, dist2(es[j], p));
  df[i] = sqrt(d2);
} // baseline version

nuwork(128) dmine(4) do(i in m) {
  def p = ps[i];
  mutable d2 = 1e38f;
  unroll(2) do(j in n1)
    d2 = min(d2, dist2(es[j], p));
  df[i] = sqrt(d2);
} // version optimized for NVidia Fermi GPUs

nuwork(128) dmine(10) do(i in m) {
  def p = ps[i];
  mutable d2 = 1e38f;
  unroll(2) do(j in n1)
    d2 = min(d2, dist2(es[j], p));
  df[i] = sqrt(d2);
} // version optimized for AMD GPUs
```

data in GPU kernels.

NUDA makes GPU programming really straightforward, once the program consists of parallel loops processing arrays. For example, array addition, traditional GPU "hello world!", is really straightforward, see Algorithm 1. In fact, **do**(...) is a loop, which can be multi-dimensional, and **nuwork**(64) is an annotation which sends the loop to which it is applied to GPU; here, 64 is the thread block size, and the total number of threads is derived from the number of iterations on the GPU. The implementation of array addition on GPU above is, in fact, near-optimal, as the problem is memory-bound, and the implementation almost saturates memory bandwidth. For other problems, however, such naïve implementations are far from optimal. In these cases, annotations performing loop transformations can be applied to optimize the code. Annotations include full loop unroll (**inline**), standard unroll (**unroll**), deep loop unroll (**dmine**), caching data in local memory (**ulocal**), loop tiling as well as other transformations.

Distance field computation is the problem of computing distances from a set (an array) of points to the border of a polygon, one distance per point. In the basic case, for each point, the distance is the minimum of distances from the polygon to each of the points in the set. There are two main approaches to distance field computation. The first one is *all-to-all* approach, in which for each point, distance to each of the edges is computed, and minimum distance is found. The second approach is to use an HBV (hierarchy of bounding volumes), e.g. BSP trees. While the second approach is asymptotically better, it is not clear outright which one is faster, since all-to-all approach allows more aggressive optimization and is more cache friendly.

For our testing, we implemented both approaches on CPU and GPU. CPU implementation was done using C++ with OpenMP and SSE intrinsics, while GPU variant was implemented using Nemerle and NUDA. For all-to-all approach, two versions are compared: the trivial version and the optimized version. Here, optimization was done using NUDA annotations without compromising readability, see Algorithm 2. The basic GPU code version, together with annotations used for optimization, is presented below. **dmine**(k) annotation does deep loop unrolling. This means creating k copies of the loop and zipping them together, including inner loop. It can also be thought of as computing k elements in a single GPU thread at the same time. For AMD, this allows benefiting from its 5-way VLIW instructions, and it also results in less global memory bandwidth for both architectures. **unroll**(k) does simple k-sized loop unrolling, and reduces looping overhead for the inner loop on NVidia GPUs.

For the second approach, we implemented HBV-assisted distance computation on GPU. We first construct a BSP tree using fixed-point (at half size) recursive space subdivision, and switching subdivision axis for subsequent subdivisions. We stop when either the maximum depth or minimum number of edges is reached, or when the overhead resulting from the subsequent subdivision is higher than a certain threshold. After that, we transform the BSP into HBV node-wise from the bottom up, by recomputing the bounding volume based on all edges belonging to a specific node. For distance computation, we use a traditional recursive algorithm. We first compute the distance from the point to both subvolumes in HBV, and then go into the nearest one. Once we have the upper estimate for the nearest distance, we prune all subvolumes with greater distance. Since not all current GPUs support recursion, we've rewritten the algorithm, first as a stack-based traversal, and then as a stack-less traversal by adding pointer-to-parent to each GPU HBV node. HBV traversal is obviously less regular than all-to-all computation, so none of the optimizations described above is used. Each GPU thread traverses HBV for one point only.

We tested the algorithms we implemented on two GPUs. For both approaches, all computations were done in single precision. The first one is NVidia Tesla C2050 (nvidia) with 3 GB of GPU RAM and 1030 GFlop/s single-precision peak performance. The second one is ATI HD Radeon 5830 (ati) with 256 MB of GPU RAM 1792 GFlop/s single-precision peak. We performed two series of experiments. Both involved computing a distance field from a uniform 2D $m \times m$-sized grid of points to a regular polygon centered at $0.5 * m$ inscribed into a circle with radius $0.4 * m$. In the first series of experiments, the number of points was fixed at 102400, and the number of vertices varied (exponentially) from 1 to 2048. In the second series, the number of vertices was fixed at 1000, while the number of points varied from 1 to 262144, also exponentially. In both cases, we measured time it took to compute the distance field for simple all-to-all, optimized all-to-all and HBV approaches. We also computed accelerations of optimized vs. baseline and HBV vs. all-to-all baseline approaches. We shall now discuss the results, presented in figures 6 and 7.

In all cases, the growth of time with the number of vertices seems to be linear. For all-to-all approach, this must be the case, while for HBV, this means that even 2000 vertices is too few to reach logarithmic growth. When the number of points varies but stays below 2K to 4K, time varies little with the number of points, as GPU is under-saturated. After that, the expected linear growth can be observed.

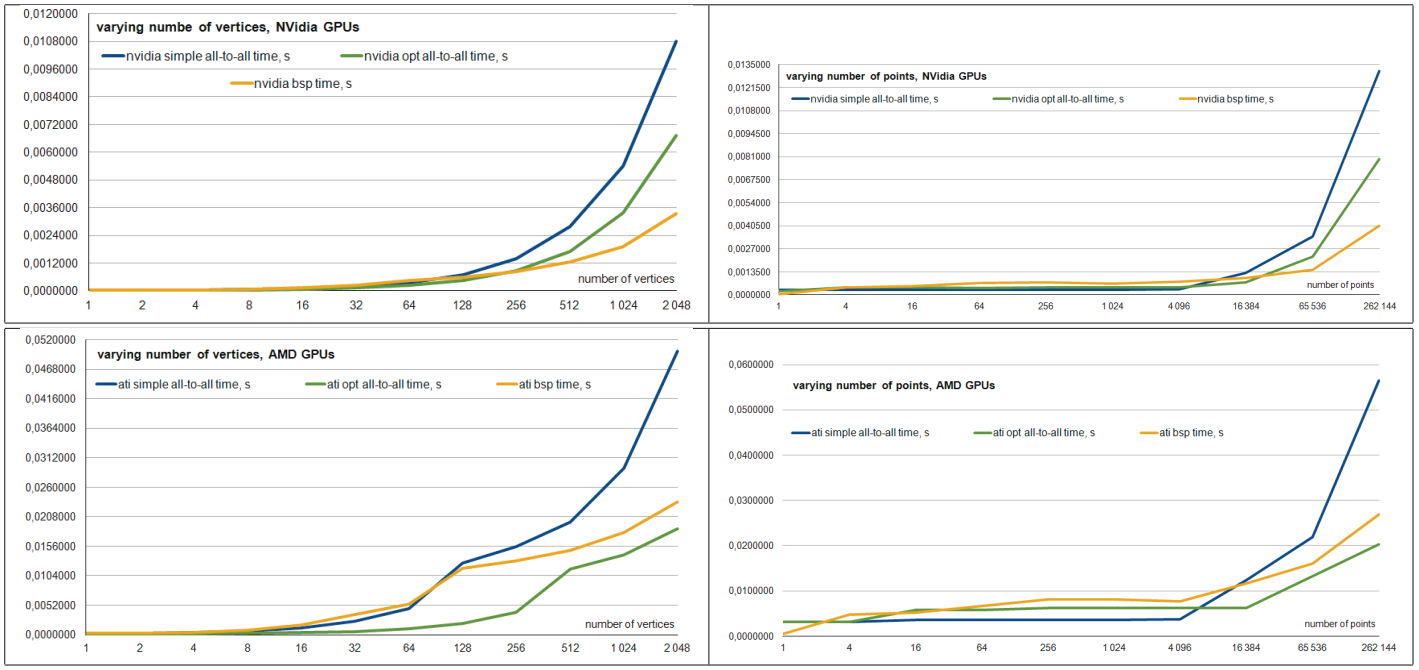

Figure 6: *Average distance field computation time for varying number of vertices, left, and points rigth. Top, NVidia; bottom, AMD.*

Obviously, for all-to-all approach, the computation time does not depend on the configuration of point field and the polygon. For HBV approach, however, time depends on the configuration of both. For ATI GPUs, HBV is always worse that optimized all-to-all for the uniform point grid. For NVidia, obviously, the acceleration increases with the number of vertices, reaching about 2 times for 2K vertices. We also tested points uniformly distributed on a circle, with the circle being close to the original polygon. Here, HBV approach can be an order of magnitude faster that optimized approach. Overall, HBV gives better acceleration for NVidia GPUs than for ATI, since NVidia Fermi GPUs have cache, and are therefore better at handling random memory accesses.

We also compared optimized all-to-all computation to simple computation. Starting with a reasonable number of points (> 4K) and almost for any number of vertices (> 4) the version optimized with annotations described above is always faster than the baseline version, with acceleration increasing with the number of points, and stopping at near 2× for both NVidia and ATI GPUs. Overall, the provided optimizations allow us to utilize GPUs efficiently, reaching about 54% peak for NVidia GPUs, and 31% peak for ATI GPUs. The times above do not include time spent on copying data (though we found it to be negligible) and on building HBV (non-negligible). Therefore, all-to-all approach can be significantly faster if the distance field is computed for each polygon only once, since it avoids the costly pre-processing step.

5 Conclusion

We analized the robustness problems of the existing distance field-based shape matching algorithms, isolating what we believe is a core issue, and proposed a new dissimilarity measure, explicitly designed to overcome it. We also presented an efficient NUDA-based GPU implementation of the distance field calculation, which makes the proposed approach computationally feasible. This is an ongoing project, with work underway to extend the approach to 3d.

Acknowledgements

This work was supported by the RFBR grant 11-04-12174-ofi-m-2011. The first author was partially supported by the contract No 07.514.12.4030. The second author was supported by CUDA Center of Excellence in the Lomonosov Moscow State University. The third author was partially supported by the contract No 13.G36.31.0002.

References

[Adinetz 2011] A.V. Adinetz. Extensible languages for GPU programming. In Proceedings of PaVT-2011. (in Russian)

[Adinetz 2012] A.V. Adinetz. Extran and NUDA Programmer's Guide. June 2012. http://sourceforge.net/projects/nuda/files/0.0.6/extran-guide.pdf/download

[Babalola 2006] K. Babalola, T. Cootes. Registering richly labelled 3d images. In: Proc. of the Int. Symp. on Biomedical Images (2006)

[Baran 2009] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic deformation transfer. Proceedings of SIGGRAPH (2009), article no 36.

[Belongie 2001] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. The 8th ICCV, Vancouver, Canada, pages 454461, 2001.

[Byrd 1995] R.H. Byrd, P. Lu, J. Nocedal, C.Y. Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM Journal on Scientific Computing, Vol. 16, No. 6. (1995), pp. 1190-1208.

[Ciesielski 2011] K.C. Ciesielski, X. Chen, J.K. Udupa and G.J. Grevera. Linear Time Algorithms for Exact Distance Transform. Journal of Mathematical Imaging and Vision, vol. 39, no. 3 (2011), 193-209.

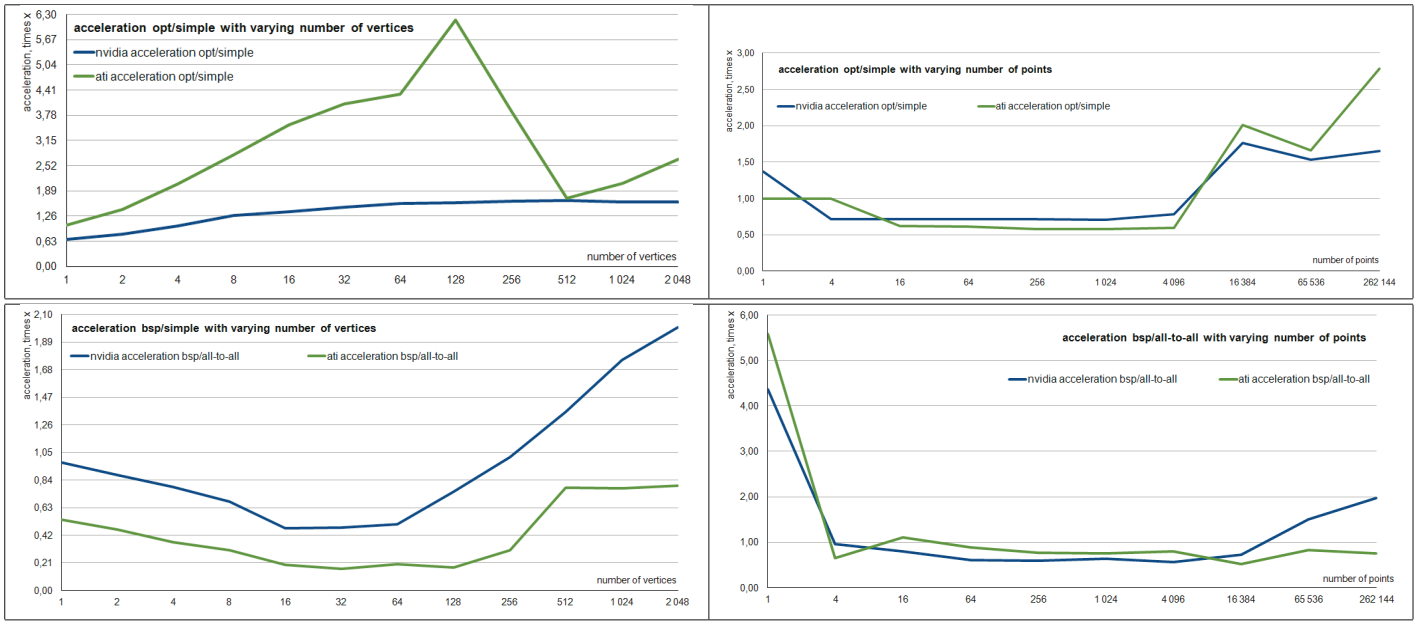

Figure 7: *Acceleration for GPU-specific optimization, top, and BSP-trees, bottom, for varying number of vertices, left, and point, right.*

[Fahmi 2008] R. Fahmi and A. A. Farag. A Novel Shape Registration Framework and Its Application to 3D Face Recognition in the Presence of Expressions. Lecture Notes in Computer Science, 2008, Volume 5359/2008, 287-296.

[Ginneken 2001] B. Ginneken, B. Romeny, and M. Viergever. Computer-Aided Diagnosis in Chest Radiography: A Survey. IEEE Transactions on Medical Imaging, Vol. 20, 2001.

[Hong 2006] B.-W. Hong, E. Prados, S. Soatto, L. Vese. Shape Representation based on Integral Kernels: Application to Image Matching and Segmentation. IEEE Conference on Computer Vision and Pattern Recognition 1 (2006) 833- 840.

[Huang 2006] X. Huang, N. Paragios, D.N. Metaxas. Shape Registration in Implicit Spaces Using Information Theory and Free Form Deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 8, Aug. 2006.

[Kilian 2007] M. Kilian, N.J Mitra, and H. Pottman. Geometric modeling in shape space. Proceedings of SIGGRAPH (2007), volume 26, pp. 64:1–64:8.

[Knott 2000] G.D. Knott. Interpolating Cubic Splines. Springer, 2000.

[Kroon 2011] D.J. Kroon. Segmentation of the Mandibular Canal in Cone-Beam CT Data. PhD thesis, University of Twente. 2011.

[Levoy 2000] M. Levoy, S. Rusinkiewicz, M. Ginzton, J. Ginsberg, K. Pulli, D. Koller, S. Anderson, J. Shade, L. Pereira, J. Davis, and D. Fulk. The Digital Michelangelo project: 3d scanning of large statues. Proceedings of SIGGRAPH (2000), pp. 131-144.

[Liu 2011] W. Liu and E. Ribeiro. A Meshless Method for Variational Nonrigid 2-D Shape Registration. Proceeding of ISVC 10, vol II, pp. 262-272.

[Maintz 1998] J.B.A. Maintz, , M.A. Viergever. A survey of medical image registration. Medical Image Analysis 2 (1998) 1–36.

[Malcolm 2008] J. Malcolm, Y. Rathi, and A. Tannenbaum. Label Space: A Multi-object Shape Representation. Lecture Notes in Computer Science, 2008, Volume 4958/2008, 185-196.

[Munim 2007] H.A.E. Munim, A.A. Farag. Shape representation and registration using vector distance functions. Proceedings of CVPR 2007.

[Paragios 2003] N. Paragios, M. Rousson, and V. Ramesh. Non-rigid registration using distance functions. Computer Vision and Image Understanding, 89(2-3):142{165, 2003.

[St-Jean 1998] P. St-Jean, A.F. Sadikot, L. Collins, D. Clonda, R. Kasrai, A.C. Evans, and T.M. Peters. Automated Atlas Integration and Interactive ThreeDimensional Visualization Tools for Planning and Guidance in Functional Neurosurgery. IEEE Transactions on Medical Imaging, Oct. 1998.

[Tsai 2003] A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, W. Grimson, A. Willsky. A shape-based approach to the segmentation of medical imagery using level sets. Transactions on Medical Imaging 22(2), 137–154 (2003).

[Veltkamp 1999] R.C. Veltkamp and M. Hagedoorn. State of the art in shape matching. Technical Report UU-CS-1999-27, University of Utrecht, 1999.

[Voronin 2009] P. Voronin, A.V. Adinetz. On computing distance fields of planar polygons. In proceedings of GraphiCon-2009. (in Russian)

[Zitova 2003] B. Zitova and J. Flusser. Image registration methods: a survey. Image and Vision Computing, 21 (11) (2003) 977–1000.

[Zhou 2005] X.S. Zhou, A. Gupta, and D. Comaniciu. An information fusion framework for robust shape tracking . IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 1, 2005.

[Zhu 1996] S.-C. Zhu and A. Yuille. Forms: a flexible object recognition and modeling system. IJCV, 20:187–212, 1996.

Technical section (Russian)

GraphiCon'2012

October 01–05, 2012
Moscow, Russia