The DBICP Project

April 2009

Jean-Baptiste Fiot
Ecole Centrale Paris
Ecole Normale Supérieure de Cachan
jean-baptiste.fiot@student.ecp.fr

Abstract

The DBICP Project studies a point based registration algorithm called the "Dual Bootstrap Iterative Closest Point" (see [1]).

This algorithm extends the classic Iterative Closest Point (ICP) algorithm, to overcome issues such as initialization sensitivity, few overlap, and unreliable matches. The innovations are made in the algorithm's structure, where the region used – the bootstrap region –, and the parametric transformation model selected are progressively "bootstrapped", meaning enlarged.

Finally, my source code is available (for free!) at [3].

1. Introduction

The *Iterative Closest Point (ICP)* algorithm is a point-based registration algorithm. It should be used when correspondences are not known and when matching based on the properties of individual points (and their surroundings) does not produce a large enough set of unique correspondences to precisely align the data.

As explained in section 2.1, ICP is an iterative minimization algorithm, and therefore requires a proper initialization. Despite the efforts and research done on initialization and efficient matching, there are some cases where initial estimates alone are not enough to ensure that ICP will converge to an accurate transformation estimate. This is the case in human retina registration for instance.

To overcome this problem, the region used – the *bootstrap region* –, and the parametric model selected for the transformation are progressively "bootstrapped" ie enlarged. The main idea is to compute an approximate transformation, and then refine it progressively.

2. Basic ICP

2.1. Algorithm outline

The ICP algorithm iterates two steps:

- a matching step
- an optimization step

The matching step computes the correspondences between the two point sets, given a transformation T.

The optimization step looks for the best parameter set θ for the transformation, given some correspondences.

2.2. Matching step

The matching step aims to find the correspondences between the two point sets. Indeed, we have not any a priori knowledge to register the two images.

This step is really basic, to each point p in the first image, we associate the closest point to T(p) in the second image point set. We can simply use the classic Euclidean distance.

2.3. Optimization step

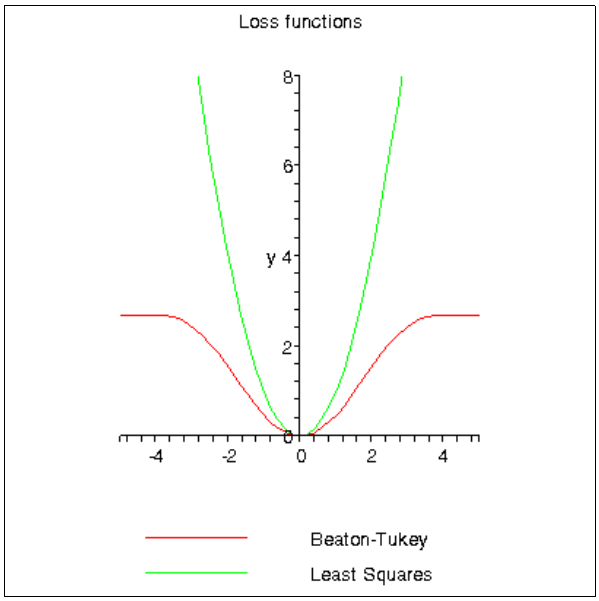
The optimization is seen as the minimization of the energy or cost function:

$$E(\theta, \sigma, C) = \sum_{p_i, q_j \in C} \rho\left(\frac{d(T_\theta(p_i), q_j)}{\sigma}\right)$$

$$with \begin{cases} \theta \;\; parameters\ of\ the\ T_\theta\ transformation \\ \{p_i, q_j\}\ points\ in\ the\ images \\ \sigma \;\; error\ scale \\ \rho \;\; loss\ function \end{cases}$$

The loss function recommended in [1] in the Beaton-Tukey, defined as:

$$\rho(u)=\begin{cases}\frac{a^2}{6}\left[1-\left(1-\left(\frac{u}{a}\right)^2\right)^3\right] & if\ |u|\le a\\ \frac{a^2}{6} & otherwise\end{cases}$$



As said in [1], using Beaton-Tukey instead of Least Squares significantly improves the results.

To solve the previous optimization problem, several methods are available:

- Iteratively Reweighted Least Squares (IRLS) (it is the method used in [1])
- Levenberg Marquardt
- Basic Gradient descent based on finite differences

To start, I have implemented a basic gradient descent (with ρ constant). At each iteration, parameters are updated with the following rule:

$$\theta_i \leftarrow \theta_i - \rho * \frac{E(\theta_i+\epsilon)-E(\theta_i)}{\epsilon} \approx \theta_i - \rho * \nabla E_i$$

(Note for E : other parameters remain constant in the finite difference)

2.4. Implementation

As any optimization algorithm, ICP and DBICP are computationally expensive. That's why I have naturally chosen C++ for my implementation. I have used the CImg library [2] to manipulate images, display and save results.

Please refer to my Google Code website [3] to get the full source code, get binaries, see videos, and much more!

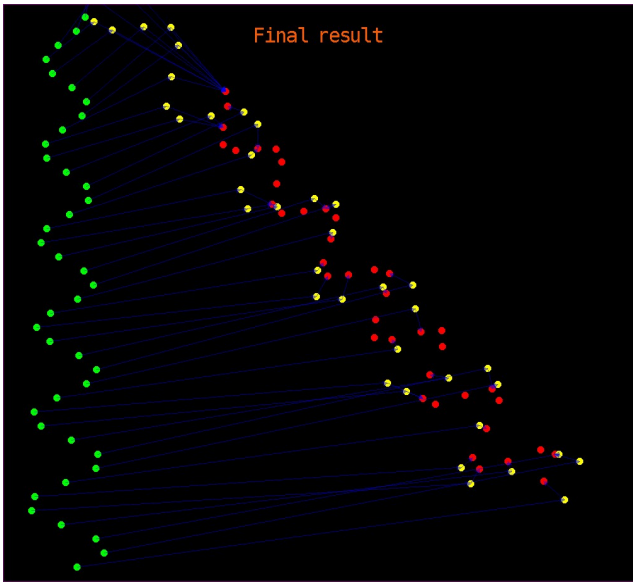
2.5. Results

In the following examples, the green dots are the original points, the red points are the target points, and the yellow points are the images of the green points via the estimated transformation. The red points are the images of the green points via an arbitrary similarity.

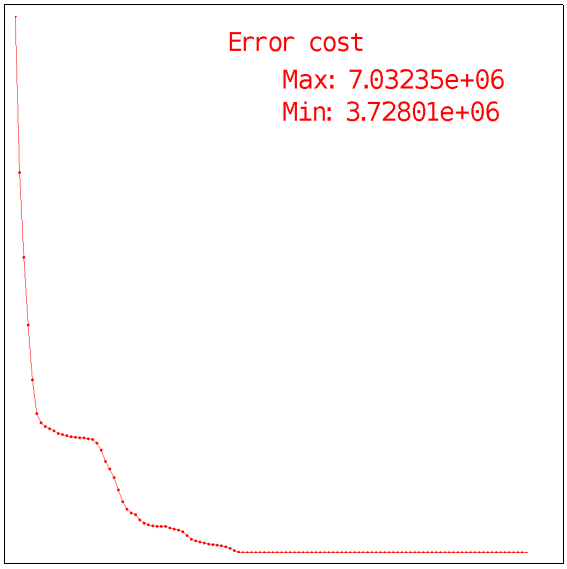
The blue arrows link each green point to its image via the transformation T_θ, and each yellow points to its corresponding target (the closest point in the red point set).

2.5.1 Algorithm stuck in local minima

In this first example, the initial transformation is the identity. In the end, the yellow structure is quite well aligned with the red structure, but the alignment is poor. The θ_{12} and θ_{21} parameters, corresponding to the translation, are too low and the scaling factor is too high, to catch up this bad translation.

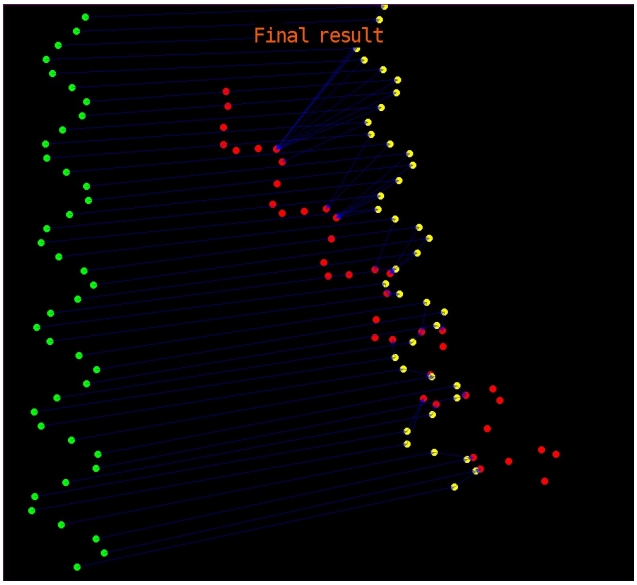


It is also interesting to have a look at the cost function during the iterative process.



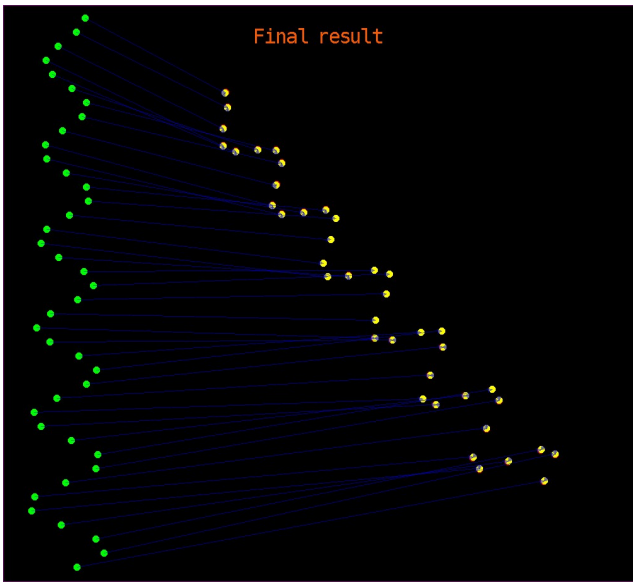
Several times, the decrease seems to vanish, and at a stretch reappears! This can be explained very simply. Indeed, during several iterations, the correspondences remain the same, and the cost decreases quickly at the beginning, and progressively at a lower speed. When the correspondences change, it gives a better optimization and thus the cost decreases quickly again! Isn't that cool? ;)

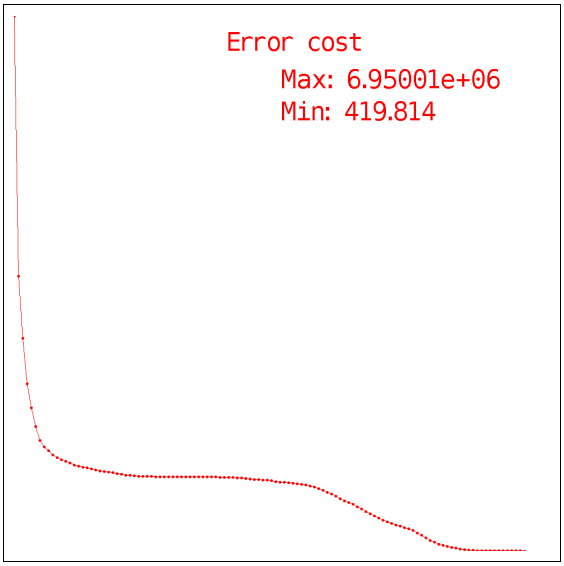
In the second example, the initial transformation is the translation from the barycenter of the green points to the barycenter of the red points. In this case, the algorithm is stuck really quickly in a local minimum, and the registration is disastrous!



2.5.2 Successful example

With a better initialization, the basic ICP can manage to find the global minimum of the cost function, which basically means the registration is great! Here is a perfect success:





2.6. Conclusion

The ICP algorithm is based on a good concept, but suffers many drawbacks:

- it is often stuck in a local minimum
- it is very sensitive to initialization
- it needs a large enough overlap
- it is sensitive to outliers

3. Dual Bootstrap ICP

3.1. Concept

As explained in the introduction, the innovation in the DBICP algorithm is a change in the structure of the basic ICP algorithm. The main idea is to compute an approximate transformation, and then refine it progressively. This refinement is done thanks to a progressive growth of the region used to compute the correspondences, and a switch in the parametric models: from low-order models to higher-order models. Using low-order models at the beginning prevent to be stuck in a local minimum.

The term *"dual-bootstrap"* refers to simultaneous growth in the bootstrap region and the transformation model order. It is applied to one or more initial bootstrap regions independently, ending with success when one bootstrap region R is expanded to a sufficiently-accurate, image- wide transformation.

3.2. Algorithm outline

- Initialization:
  - Bootstrap region
  - Initial transformation
- Iterations:
  - Compute correspondences
  - Optimize the transformation
  - Bootstrap the region
  - Bootstrap the model

In section 2, we explained how the correspondences are found, and how the transformation is optimized. In the following sections 3.3 and 3.4, we will explain how the region and the model are bootstrapped.

3.3. Region Bootstrap

3.3.1 Concept

Instead of being computed on the complete picture, the algorithm is executed in a restricted area called *"bootstrap region"*. This region, R, is defined as the image region over which the transformation is considered accurate. It determines a set of points to which the ICP algorithm is applied. Initially, R is small, surrounding the points used to generate an initial estimate. The bootstrap region is gradually grown (bootstrapped) to an image-wide, accurate transformation estimate.

A growth parameter has to be chosen for this part.

3.3.2 Results

I have implemented a basic region bootstrap, with expanding the bootstrap region where the error decrease vanishes. However, the results are not really relevant on these simulated examples. It should be interesting to try it on real data, with outliers to deal with.

3.4. Model Bootstrap

3.4.1 Concept

Rather than using a single, fixed transformation model, different models are used in different bootstrap regions, starting from a simple model for the initial bootstrap region and gradually evolving to a higher-order model as the bootstrap region grows to describe the entire data set.

Model selection techniques, which depend fundamentally on the parameter estimate covariance matrix, are used to automatically select the transformation model for each bootstrap region.

The various parametric transformation models used are: similarity, affine, reduced quadratic, and quadratic.

To each point p = (x,y), we associate the vector X(p) = (1,x,y,x², x*y, y²). We represent the various parametric transformations using the following matrices:

Transfo	Matrix	DoF
Similarity	$\begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & 0 & 0 & 0 \\ \theta_{21} & -\theta_{13} & \theta_{12} & 0 & 0 & 0 \end{pmatrix}$	4
Affine (not used)	$\begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & 0 & 0 & 0 \\ \theta_{21} & \theta_{22} & \theta_{23} & 0 & 0 & 0 \end{pmatrix}$	6
Reduced Quadratic	$\begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} & 0 & \theta_{14} \\ \theta_{21} & -\theta_{13} & \theta_{12} & \theta_{24} & 0 & \theta_{24} \end{pmatrix}$	6
Quadratic	$\begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} & \theta_{16} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} & \theta_{25} & \theta_{26} \end{pmatrix}$	12

The image points are computed using q=M*X(p), except for the reduced quadratic: q=M*X(p-p_0) with p_0 being the center of the region of overlap between the images .

Obviously, a high degree of freedom (DoF) is penalized, to prevent the direct use of complex models.

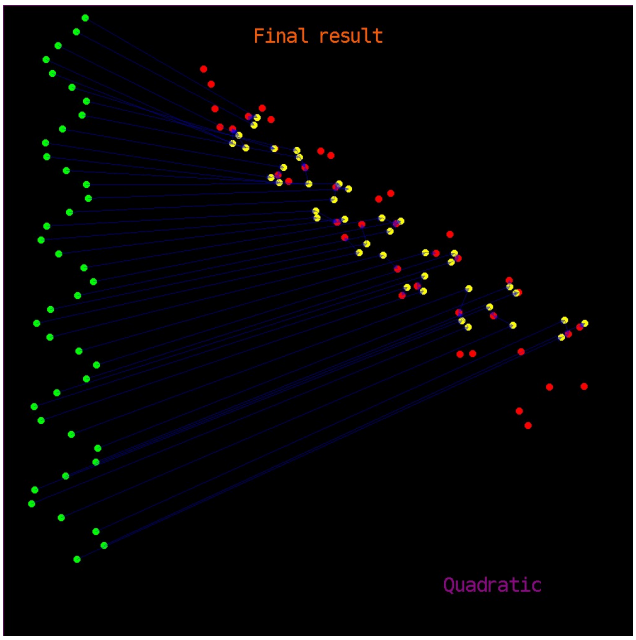
The selection criterion has the form:

$$E(T_\theta, \sigma, C) + P * DoF$$
with P the penalization of the DoF

3.4.2 Results

Here is an example of the effect of Model Bootstrapping.

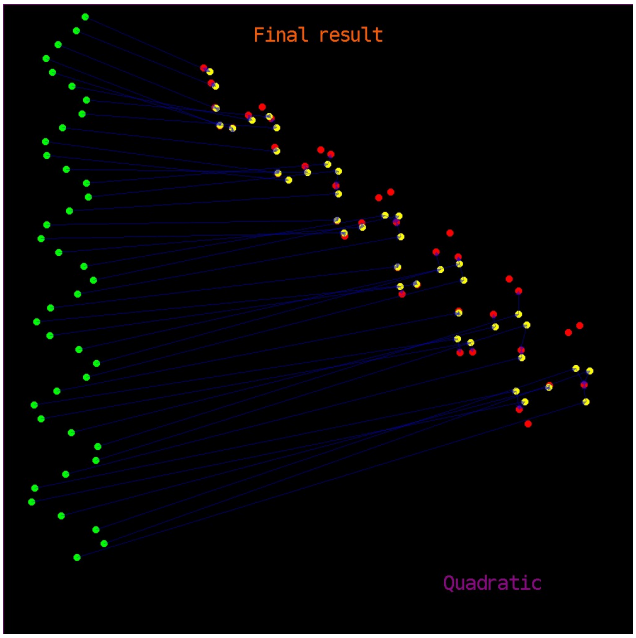
This first picture is the result without Model Bootstrapping. A quadratic function has been used to move the green points to the red points. In the 1st case, the algorithm is forced to use quadratic function in the optimization process. Here is the result:



In the second case, we use the same quadratic function to move the green points. However, in this case, we let the algorithm select automatically the parametric model it used.

At the beginning, it starts with similarities, find an approximate result, and then switch to quadratic functions, to refine this result.

Here is the final result in this case:



The result is not perfect yet, but it clearly demonstrate model bootstrap's potential.

4. Conclusion and further work

The Dual Bootstrap Iterative Closest Point presents a nice concept in its innovations of the ICP algorithm's structure.

One can find in [3] an efficient and flexible C++ implementation. However, the enthusiasm is pretty mild on simulated registration tests. It would be interesting to test it on real data, try with descriptors, with other optimization methods, other loss functions, etc.

References

[1] C. V. Stewart, C-L Tsai, B. Roysam. The Dual-Bootstrap Iterative Closest Point Algorithm with Application to Retinal Image Registration. International Journal of Computer Vision. June 2002.

[2] The CImg library. http://cimg.sourceforge.net/

[3] The DBICP Project Home Page http://code.google.com/p/dbicp