CHEMNITZ UNIVERSITY OF TECHNOLOGY

Faculty of Natural Sciences

Institute of Physics

Bachelor Thesis

Image Alignment

Katharina Wagner

Chemnitz, May 31, 2006

Supervisor: Prof. K. H. Hoffmann

Wagner, Katharina
Image Alignment
Bachelor Thesis, Faculty of Natural Sciences
ChemnitzUniversity of Technology, May 2006

Acknowledgment

Abstract

Aligning two images by point to point correspondence is a hard optimization problem. It can be solved using τ-Extremal Optimization or with a modification of this method called Fitness threshold accepting. In this work these two methods are tested and compared to see wether one of the methods should be prefered for image alignment. Since real image data is almost always noisy the performance of the methods under conditions like noisy and outlying data is analysed too.

Contents

List of Figures

List of Tables

1 Introduction

The problem of aligning two or more images occurs in many areas. E.g. in the medical image analysis it is often necessary to compare images that are taken by different sources, at diverse times or from various viewpoints. Therefore it is useful to find the transformation between these images and to align them. So one can detect changes more easily.

Another field of application is biometric data analysis. The image of the face or iris of a person has to be aligned to the reference image so that it can be recognized by a computer.

Autonomous machines like robots which recognize barriers depend also on image registration methods. The two images for a stereo view have to be aligned to recognize e.g. the shape of the barrier.

The aligning of two images can be done by point-to-point correspondence. The points of interest have to be extracted from the images, e.g. using an algorithm like the combined corner and edge detector proposed by Harris [**?**]. Then these points have to be matched in the correct order which is a combinatorial optimization problem. Especially for large numbers of points this is a computationally intensive task. To solve this problem in a fast way, Extremal Optimization (EO) can be used. EO is a powerful stochastic optimization method inspired by the concept of Self-Organized Criticality [1]. In contrast to other optimization methods like Simulated Annealing, EO leads very fast to near optimal solutions. As EO is based on minimizing a cost function it can lead into loacal minima of this function, unable to leaf them again. To avoid this a variation of EO, called τ-EO was introduced. It improves the search of the best solution a lot.

2 Point correspondence using Extremal Optimization

2.1 Extremal Optimization

Extremal Optimization [1] is a stochastic optimization method inspired by the Bak-Sneppen model of Self-Organized Criticality [**?**]. It searches the space $\Omega = \alpha$ which consists of all possible solutions α in order to find the best solution α_{best}. The search is based on minimizing a cost function $E(\alpha)$ which characterizes the quality of the solution, i.e. the optimal solution is equivalent to the smallest value of E.

Every state α consists of a finite number of degrees of freedom, which each can be assigned a fitness $\lambda \in [0,1]$. The basical form of EO performs a random walk through Ω in order to find the minimum of the cost function by updating the degree of freedom with the worst fitness in each step. But this random walk can lead into local minima of the cost function which can not be left again.

To avoid this sticking τ-EO was introduced. In this algorithm the degrees of freedom are ranked according to their fitness. The search is improved by not updating the degree of freedom with the worst fitness, but by choosing a degree of freedom using a time indipendent probability distribution $P \propto k^{-\tau}$ over the ranks k. So if τ is very small, near zero, the probability distribution gets similar to a uniform distribution over all fitness ranks. For $\tau \to \infty$ the first rank is chosen always and the other ones never which is equivalent to the basic EO. Well performing values for τ are shown in chapter 3.1.

Instead of the τ-probability distribution Heilmann, Hoffmann, and Salomon [2] introduced the Fitness Threshold Accepting distribution. It is an uniform distribution over the fitness ranks up to a certain threshold. In [3] it was proofed that such a distribution is the best distribution over EO fitness ranks. But it is only the optimal choice if the right threshold is chosen in each step of the algorithm. Its performans with fixed, adaptiv or randomly changing thresholds is shown in chapter 3.2

Figures 2.1 and 2.2 show the τ-EO and the FTA probability distributions over the fitness ranks. The τ-EO distribution is plotted for $\tau = 1$ and $\tau = 2$ which shows that the probability of the first ranks changes most and that the higher ranks are hardly affectet by a finite variation of τ. The uniform distribution for FTA is plotted for the thresholds 4 and 6.

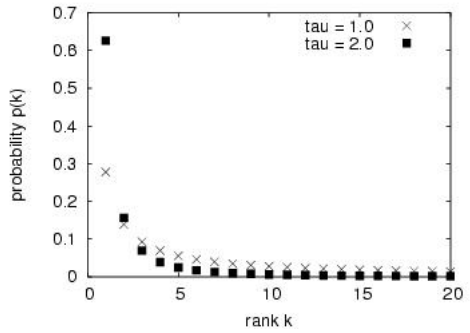

Figure 2.1: τ-EO probability distribution

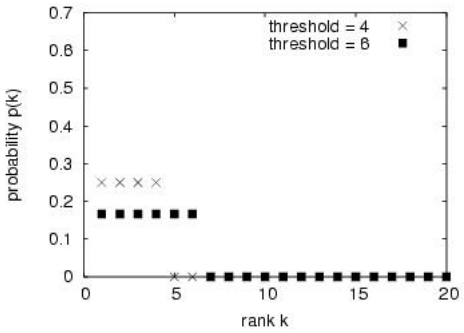

Figure 2.2: FTA probability distribution

2.2 Algorithm description

The points of interest have to be extracted from the two images whose transformation should be computed. This can be obtained by using point extraction methods like e.g. the Harris detector [**?**]. The sets P_1 and P_2 contain the points $(y_1, y_2) \in$ Image 1 and $(x_1, x_2) \in$ Image 2 in the following way.

$$P_1 = \begin{pmatrix} y_1(1) & y_2(1) \\ \vdots & \vdots \\ y_1(j) & y_2(j) \\ \vdots & \vdots \\ y_1(n_1) & y_2(n_1) \end{pmatrix} \text{ and } P_2 = \begin{pmatrix} x_1(1) & x_2(1) \\ \vdots & \vdots \\ x_1(i) & x_2(i) \\ \vdots & \vdots \\ x_1(n_2) & x_2(n_2) \end{pmatrix} \tag{2.1}$$

There correspondence between these point sets is stored in the match matrix

$$M = \begin{pmatrix} m(1,1) & \cdots & m(1,j) & \cdots & m(1,n_1+1) \\ \vdots & \ddots & \vdots & & \vdots \\ m(i,1) & \cdots & m(i,j) & \cdots & m(i,n_1+1) \\ \vdots & & \vdots & \ddots & \vdots \\ m(n_2+1,1) & \cdots & m(n_2+1,j) & \cdots & m(n_2+1,n_1+1) \end{pmatrix} \tag{2.2}$$

which is defined with an extra column and row representing outliers. Outliers are points that do not have a point match.

The match matrix element $m(i, j)$ equals one if the point $X_i \in P_2$ and $Y_j \in P_1$ are supposed to correspond. All other elements of M in the jth column and ith row equal zero. If the point $X_i \in P_2$ is an outlier, the element $m(i, n_1 + 1)$ equals one and analog if $Y_j \in P_1$ is an outlier the element $m(n_2 + 1, j)$ equals one. It is assured that the assignment is unique by the condition that the sum of every row and column of the match matrix has to equal one, except the $(n_2 + 1)$th row and the $(n_1 + 1)$th column where the outliers are stored.

Out of the matched points an affine transformation between image 1 and image 2 can be computed using a least of squares method. The transformation A_j is of the form

$$A_j = X_i = LY_j + T \tag{2.3}$$

or

$$\begin{pmatrix} a_1(j) \\ a_2(j) \end{pmatrix} = \begin{pmatrix} x_1(i) \\ x_2(i) \end{pmatrix} = \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} y_1(j) \\ y_2(j) \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \tag{2.4}$$

where T represents the translation vector and L the transformation matrix including rotation and expansion or contraction.

Since the computed transformation does not equal the exact transformation as long as the points are not matched correctly, there can be defined an error vector ε_j for each point match (X_i, Y_j). Its norm equals the distance between the point X_i and the transformation A_j of the point Y_j.

$$\begin{pmatrix} \varepsilon_1(j) \\ \varepsilon_2(j) \end{pmatrix} = \begin{pmatrix} x_1(i) \\ x_2(i) \end{pmatrix} - \begin{pmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{pmatrix} \begin{pmatrix} y_1(j) \\ y_2(j) \end{pmatrix} - \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \tag{2.5}$$

This error vector is used to define a fitness value λ_j for each point Y_j:

$$\lambda_j = \begin{cases} 0 & \text{if the point is an outlier} \\ 1 - \frac{|\varepsilon_j|}{\max\limits_k |\varepsilon_k|} & \text{otherwise} \end{cases} \tag{2.6}$$

So a point that already has a good match gets a fitness value near one. Points with bad matches or outliers get fitness values near zero.

Another measure, introduced by Meshoul and Batouche [4] is the closeness measure. Since they defined the closeness using a term they did not explain completely, I reduced it to the following term:

$$closeness(X_i, Y_j) = d(X_i, A_j) \tag{2.7}$$

where d is the Euclidean distance and A_j is the affine transformation 2.3 of Y_j. The closeness measure is needed to improve the choice of a new corresponding point for the point X_i without performing a random walk through the search space. With the help of the closeness measure points whose transformations are localized near the point X_i are preferred.

The error term is also needed to define the cost function E which characterizes the quality of the solution:

$$E(M, \varepsilon) = \sum_{j=1}^{n_1} |\varepsilon_j|^2 + \alpha \left(\sum_{i=1}^{n_2} m(i, n_1 + 1) + \sum_{j=1}^{n_1} m(n_2 + 1, j) \right) \tag{2.8}$$

The value of the cost function increases if there are more outliers and if the errors of the matches are big. The aim of the optimization is to minimize these cost function. That means solutions with less outliers and small errors are preferred in contrast to solutions with many outliers and big errors.

As initial guess of the match matrix M I chose the identity matrix and set the element $m(n_2+1,n_1+1)$ to zero. Starting from this correspondence the algorithm performs the following way:

Input Two sets of points P_1 and P_2.
Generate the initial match matrix M and compute the initial transformation A, which add up to the solution $\alpha = (M,A)$. Set $\alpha_{best} = \alpha$.

Repeat
Rank the points of P_1 according to their fitness λ_j.
Select a point Y_j from P_1 according to its fitness rank k_j using τ-EO or FTA.
If the selected point is an outlier
Then Rank the points of P_2 according to their closeness to the transformation A_j.
Select a point X_i from P_2 according to its closeness rank k_i using τ-EO or FTA. If the point X_i already has a corresponding point Y_l, the point Y_l becomes an outlier.
Match the points X_i and Y_j by updating the match matrix.
Else The match pair (X_i,Y_j) is split into two outliers.
End If
Compute the new transformation A using the updated match matrix. Set $\alpha = (M,A)$.
If $E(\alpha) < E(\alpha_{best})$
Then Set $\alpha_{best} = \alpha$.
End If
Until termination criterion is achieved.

Output $\alpha_{best} = (M,A)$.

3 Experimental results

To test and compare the performance of the FTA- and the τ-EO-algorithm I used synthetically created points. That means I generated the points of set P_1 randomly and transformed them using a given transformation A into the points of set P_2. Then the points of set P_2 were mixed to avoid a trivial correspondence to the points in P_1. An advantage of these synthetically generated points is that I know the exact transformation between the two sets of points. This knowledge makes it very easy to figure out exact solutions.

To define a good termination criterion is difficult, because for real images one does not know the transformation. So the termination criterion cannot be that the right transformation was found. It is also not possible to to define a certain threshold for the cost function so that the calculation was stopped if the cost function gets smaller than the threshold. Sometimes the cost function converges zero for exact solutions but sometimes the right solution is found although there are still outliers. These outliers cause a higher value of the cost function which is nevertheless the smallest value in many calculations. So one could not define a certain value that has to be fallen below. To avoid these problems I set the number of iteration steps to fixed values which can be seen in Table 3.1. This value depends on the number of points n in P_1. Since the number of combinations in the match matrix M rises with an increasing number of points the number of iteration steps has to increase too. If the numbers of points from both images are equal, i.e. $n_1 = n_2 = n$, the number of assignments equals $n!$.

Since the calculation time rises with increasing number of calculation steps and also with increasing number of points, at least the number of calculation steps has to be limited. Especially in case of running the algorithm 100 times and in case of high numbers of points, it has to be done.

number of points (n)	number of assignments ($n!$)	number of iteration steps
5	120	100
10	3628800	1000
20	2.43e+18	4000
40	8.16e+47	20000
60	8.32e+81	20000
>60		20000

Table 3.1: Number of possible assignments and number of iteration steps according to the number of points n

To compare the performance of the different algorithms I counted the exact or almost exact solutions which were found by the algorithms during 100 runs, i.e. how often was the given transformation found as the best transformation. Each of these runs was started with different random points but the same points were taken for each algorithm. So the same transformations for the same points had to be found by the different algorithms. Since I created the points of set P_2 by transforming the points of set P_1, both sets contain the same number of points. Thus there were no outliers that had to be considered.

For the testing I defined the following transformations which consist of a translation defined in T and a rotation combined with a scaling in the matrix L. The rotation is defined by the rotation angle φ and the scaling by the scaling factor s. Then the matrix L can be written as:

$$L = s \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \tag{3.1}$$

The first transformation A_1 is a rotation about 90 degrees combined with an expansion by 2 and a translation:

$$L_1 = \begin{pmatrix} 0 & 2 \\ -2 & 0 \end{pmatrix} \quad \text{and} \quad T_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \tag{3.2}$$

The second transformation A_2 is only an expansion by 1.5 without any rotation or translation:

$$L_2 = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix} \quad \text{and} \quad T_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{3.3}$$

3.1 Performance of τ-EO depending on τ

The number of exact solutions during 100 runs of the algorithm depends very much on the value of τ. Meshoul and Batouche [4] recommended a value between 2.0 and 2.5 for τ, but I found that there is no value of τ that is the best in all cases. The value of τ where the most solutions are found depends on the number of points given in the images. In figures 3.1 and 3.2 the number of exact solutions during 100 runs is plotted against τ. One can see that on the one hand for 5 points (figure 3.1) there is no perfect tau. It can vary from 1.7 to 3.0 with only less influence on the number of exact solutions. On the other hand for 40 points (figure 3.2) the best value for τ is 1.4 which produces about 63 exact solution during 100 runs. Small variations of this value lead to much less solutions.

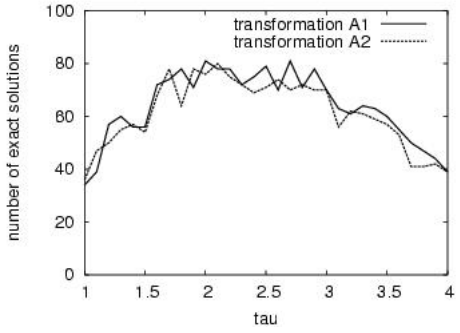

Figure 3.1: Number of exact solutions depending on τ for 5 points.

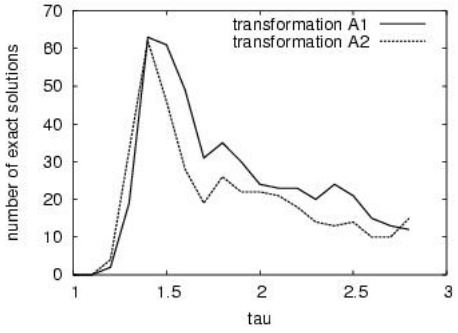

Figure 3.2: Number of exact solutions depending on τ for 40 points.

In table 3.2 you can see which values of τ are the best according to different numbers of points extracted from the images. The values and the interval of the optimal τ get smaller with increasing numbers of points. In the third column I added the values of τ that I used for the following testing of the algorithm.

number of points	τ_{best}	τ used for testing
5	$1.7 \leq \tau_{best} \leq 3.0$	2.3
10	$1.6 \leq \tau_{best} \leq 2.3$	1.9
20	$1.6 \leq \tau_{best} \leq 1.8$	1.7
40	$\tau_{best} = 1.4$	1.4
60	$\tau_{best} = 1.4$	1.4
80	$\tau_{best} = 1.7$	1.7

Table 3.2: Best values for τ according to the number of points in P_1

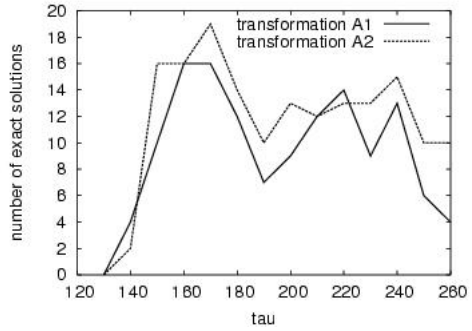

Figure 3.3: Number of exact solutions depending on τ for 80 points.

Since Meshoul and Batouche extractet more than 70 points from their images there recommended values for τ do not correspond to mine. I found that the best values for τ for more than 40 points lie around 1.5 and not as Meshoul and Batouche found it between 2.0 and 2.5. For example for 80 points the best τ is 1.7. The number of exact solutions for τ between 2.0 and 2.5 is not much smaller as you can see in figure 3.3 but $\tau = 1.7$ works better.

The number of exact solutions found during 100 runs decreases with increasing number of points. Since the number of combinations in the match matrix increases very fast with a growing number of points (table 3.1) it takes more and more calculation steps to find the exact solution. But more calculation steps cause a growing calculation time. So the number of calculation steps has to be limited which again leads to less solutions.

In the tables 3.3 and 3.4 one can see the results of the simulations using τ-EO for the transformations A_1 and A_2. For each exact solution the best value of the cost function, E_{best}, and the corresponding calculation step h_{best} were stored. Then the mean and the standard deviation for these two variables were calculated.

The average of E_{best} increases with increasing number of points. That is the number of exact solutions containing outliers increases, because the exact transformation can only be calculated from correct matched points. So the first part of the cost function containing the errors of the matches approaches zero whereas the second part containing the outliers grows with increasing number of points.

For less points ($n \leq 20$) the mean of the calculation steps h_{best} lies a bit beneath the half of the steps made (compare table 3.1), i.e. more than a half of the exact solutions were found in the first half of the calculation. That shows that the number of steps made is big enough in most cases.

For more than 20 points the mean of h_{best} lies above the half of the calculation steps made. So in these cases it would be better to make more steps to find the right solution but this would also mean that it would take much more time.

n	solutions	mean of E_{best}	std of E_{best}	mean of h_{best}	std of h_{best}
5	72	0.3333	1.1133	47.1111	27.2627
10	82	1.4634	3.6925	422.3415	273.2362
20	68	0.8895	3.8988	1.7319e+03	1.1925e+03
40	63	36.3773	9.6537	1.2747e+04	5.7594e+03
60	29	25.4018	7.9454	1.2014e+04	5.2534e+03

Table 3.3: Results of simulations using τ-EO for the transformation A_1

n	solutions	mean of E_{best}	std of E_{best}	mean of h_{best}	std of h_{best}
5	72	0.2778	1.0240	44.6944	29.1811
10	71	0.7019	2.3789	421.2778	279.4088
20	66	1.5830	4.1147	1.9108e+03	1.1308e+03
40	62	36.0845	10.8264	1.2092e+04	5.5663e+03
60	22	27.7720	6.0105	1.2746e+04	4.9529e+03

Table 3.4: Results of simulations using τ-EO for the transformation A_2

3.2 Performance of FTA depending on the threshold

The number of exact solutions during 100 runs depends a lot on the threshold used for the FTA-algorithm. I tested different modifications of this algorithm. The first one was with a fixed threshold (subsection 3.2.1), i.e. one certain threshold was chosen and it was never changed during the calculation. Another one was with an adaptive threshold that depends on the number of outliers in each calculation step (subsection 3.2.2). The third type of modification was one where the threshold was randomly set to a certain value in each step (subsection 3.2.3).

3.2.1 FTA with fixed threshold

In the modification with the fixed threshold one certain value can be detected with which the best results can be achieved. In figures 3.4 and 3.5 the number of exact solutions during 100 runs is plotted against the fixed threshold. The maximum of solutions was found at a threshold of 2 for 5 points and at a threshold of 5 for 40 points. These best fixed thresholds are hardly influenced by the type of transformation. The number of solutions found depends on the transformation but the best thresholds are the same for both of the tested transformations.

As you can see in table 3.5 the optimal fixed threshold depends on the number of points extracted from the images. For large numbers of points the optimal threshold does not change

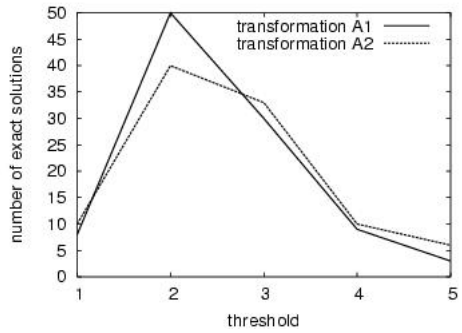

Figure 3.4: Number of exact solutions depending on the threshold for 5 points.

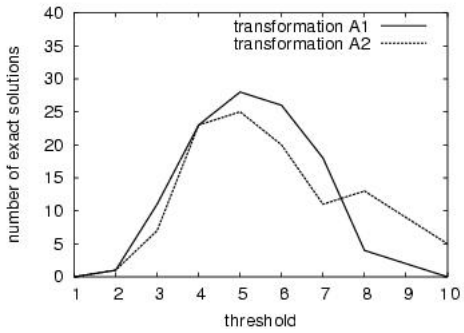

Figure 3.5: Number of exact solutions depending on the threshold for 40 points.

anymore, it remains 5. This seems to make sense because when τ-EO is used the probability that one of the first few ranks is chosen is much bigger than for higher ranks where the probability converges 0.

number of points	optimal threshold
5	2
10	3
20	4
40	5
60	5
>60	5

Table 3.5: Best fixed thresholds according to the number of points in P_1

In the tables 3.6 and 3.7 one can see the results of the simulations using FTA with a fixed threshold for the transformations A_1 and A_2. For these simulations the optimal thresholds shown in table 3.5 were used. Like for the τ-EO algorithm the mean of the cost function increases with increasing number of points. That is FTA also finds more exact solutions containing outliers for higher numbers of points.

The mean of the iteration steps that were needed to find the exact solution lies beneath one half of the steps made only for 5 points. For more points it lies above the first half of the steps made. Thus on average it takes more time to find the correct solution with the algorithm using FTA with fixed threshold than with the algorithm using τ-EO.

n	solutions	mean of E_{best}	std of E_{best}	mean of h_{best}	std of h_{best}
5	50	1.2	1.85164	39.1	27.1212
10	51	4.38958	3.97017	550.157	255.06
20	52	10.3242	8.69552	2787.88	948.493
40	28	35.3378	15.4241	12641.1	6044.92
60	14	45.0661	13.6717	12865.5	6857.1

Table 3.6: Results of simulations using FTA with fixed threshold for the transformation A_1

n	solutions	mean of E_{best}	std of E_{best}	mean of h_{best}	std of h_{best}
5	40	0.8	1.62038	36.975	22.0378
10	44	4.78856	3.49364	503.773	310.567
20	55	8.42251	3.86423	2579.89	1090.93
40	25	28.1742	11.2963	15547	4387.47
60	10	34.4067	13.3833	14247.7	4686.71

Table 3.7: Results of simulations using FTA with fixed threshold for the transformation A_2

3.2.2 FTA with adaptive threshold

The adaptive threshold was defined as follows.

$$threshold = \sum_{i=1}^{n_2} m(i, n_1 + 1) + 1 \tag{3.4}$$

It represents the number of outliers plus one. Thus almost always one of the outliers is updated in each calculation step. The performance of the threshold during the calculation is shown in figures 3.6 and 3.7 for 20 points on the left side and for 60 points on the right side. One can see that the threshold is much smaller than the number of points. Thus the number of outliers is small too which confirms that the algorithm prefers solutions with less outliers.

3.2.3 FTA with random threshold

In this modification of FTA the threshold was set to different values randomly. E.g. for 10 points, where 3 is the best fixed threshold, one of the thresholds 2, 3 and 4 was chosen in each step. Each of these thresholds was selected with a different probability.

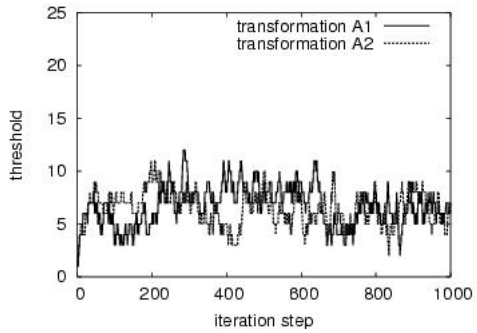

Figure 3.6: Development of adaptive threshold for 20 points.

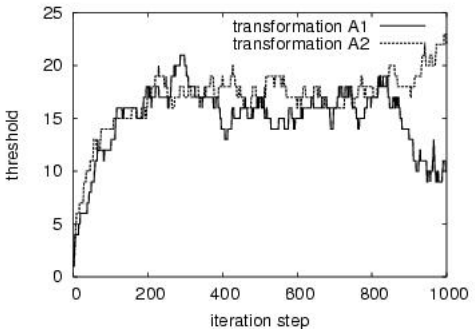

Figure 3.7: Development of adaptive threshold for 60 points.

n	solutions	mean of E_{best}	std of E_{best}	mean of h_{best}	std of h_{best}
5	37	2.39275	1.97518	52.3784	28.839
10	43	7.78914	5.76548	483.535	311.933
20	38	13.6068	11.2439	2071.21	949.264
40	26	47.8067	20.3588	11456.7	6031.93
60	0	0	0	0	0

Table 3.8: Results of simulations using Fitness Threshold Accepting with adaptive threshold for the transformation A_1

3.3 Comparison of FTA and τ-EO

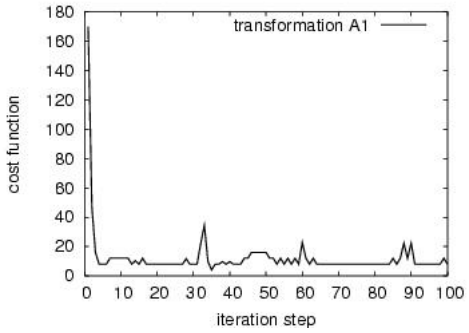

Figure 3.8: Development of the cost function using τ-EO for 5 points.

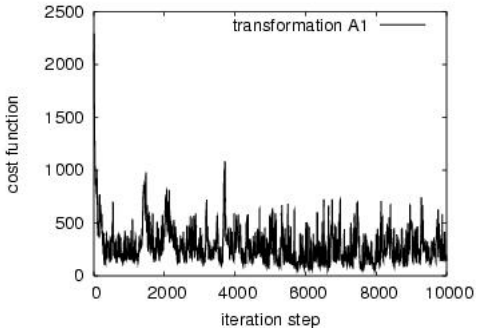

Figure 3.9: Development of the cost function using τ-EO for 40 points.

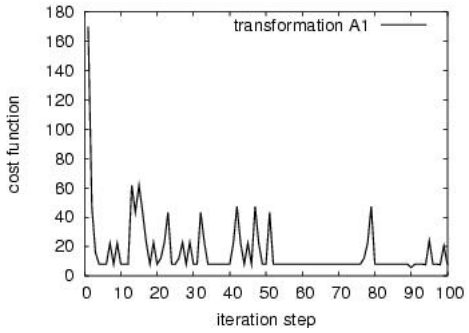

Figure 3.10: Development of the cost function using FTA with fixed threshold for 5 points.

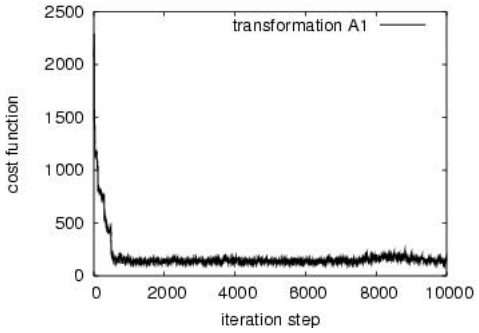

Figure 3.11: Development of the cost function using FTA with fixed threshold for 40 points.

n	solutions	mean of E_{best}	std of E_{best}	mean of h_{best}	std of h_{best}
5	33	2.06967	2.02124	52.1515	27.3669
10	40	4.59753	4.30046	568.65	300.391
20	43	12.228	10.6618	2319.07	1136.45
40	39	37.1982	18.9434	11281.4	4882.74
60	9	76.7581	15.7181	12644.3	6618.69

Table 3.9: Results of simulations using Fitness Threshold Accepting with adaptive threshold for the transformation A_2

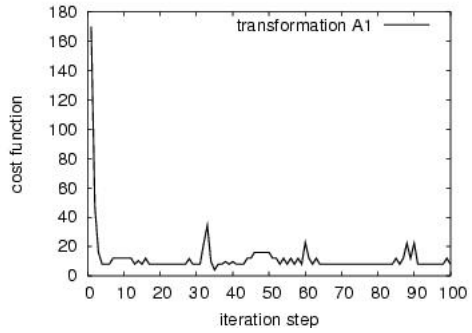

Figure 3.12: Development of the cost function using FTA with adaptive threshold for 5 points.

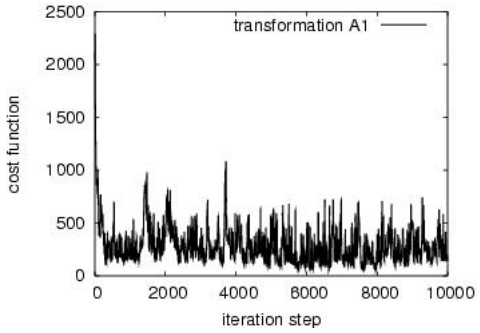

Figure 3.13: Development of the cost function using FTA with adaptive threshold for 40 points.

Bibliography

[1] S. Boettcher and A. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119:275–286, 2000. research note.

[2] K. H. Hoffmann, F. Heilmann, and P. Salamon. Fitness Threshold Accepting over extremal optimization ranks. *Phys. Rev. E*, 70(4):046704–1 – 046704–6, 2004.

[3] F. Heilmann, K. H. Hoffmann, and P. Salamon. Best possible probability distribution over Extremal Optimization ranks. *Europhys. Lett.*, 66(3):305–310, 2004.

[4] S. Meshoul and M. Batouche. Robust point correspondence for image registration using optimization with extremal dynamics. 2449:330–337, 2002.

Selbstst"andigkeitserkl"arung

Hiermit erklï¿½e ich, daï¿½ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sï¿½tliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrcklich als solche gekennzeichnet.

Chemnitz, den May 31, 2006

_____

Katharina Wagner