TELKOMNIKA, Vol.10, No.5, September 2012, pp. 1040~1050
e-ISSN: 2087-278X
accredited by DGHE (DIKTI), Decree No: 51/Dikti/Kep/2010

3D Corresponding Control Points Estimation using Mean Shift Iteration

Shan-shan Fan, Xuan Yang
College of Computer Science and Software Engineering, Shenzhen University,
Shenzhen, Guangdong, 518060, China
e-mail: shanfshan@hotmail.come-mail

Abstract

Mean shift algorithm is widely used in 2D images. In this paper a novel 3D corresponding control points estimation using mean shift algorithm is proposed. This algorithm is not a simple extension from 2D to 3D, but computes the probability density function in each slice of the search region and connects them into a whole density function smoothed by Gaussian function. And then we calculate and compare Bhattacharyya coefficients to determine a new location of the trace point. A cylinder instead of ellipsoid is utilized as the search region to improve tracking accuracy. Also three revising methods different from the direct round-off way are proposed to modify the floating trace point. Experiment demonstrates the feasibility of this 3D mean shift algorithm and the effectiveness of the three revising methods.

Keywords*: mean shift, image registration, corresponding landmark*

1. Introduction

Image registration can be classified mainly into two categories, intensity-based and feature-based methods [1]. Intensity-based methods compare intensity patterns in images through correlation metrics, while feature-based methods find correspondence between image features such as points, lines, and contours. With the high-speed and real time characteristic, feature-based, especially point-based, methods play an important role in image registration. To find corresponding points in images, the mean shift algorithm is used. Mean shift, first proposed by Fukunaga et al. [2] in 1975, is a non-parametric statistical method for seeking the nearest mode of a point sample distribution. This algorithm is a simple iterative procedure that shifts each data point to the average of data points in its neighborhood. Yizong Chenng [3] generalized the original mean shift algorithm in three aspects. That generalization broadened its range of application in many fields. Comaniciu et al. [4, 5] applied mean shift algorithm in the analysis of complex multimodal feature space, high quality edge preserving filtering and image segmentation, and discussed its convergence conditions. Also Comaniciu [6] utilized mean shift in real-time tracking of non-rigid objects, where the Bhattacharyya coefficient between the target and candidate models was regarded as a metric, and that is what our 3D mean shift algorithm is based on. Collins [7] adapted Lindeberg's theory of feature scale selection based on local maxima of differential scale-space filters to the problem of selecting kernel scale for mean shift blob tracking. Given a new video frame Avidan [8] used a classifier to test the pixels and formed a confidence map, and finally used mean shift [9] to find the peak of the map where they believed was the position the object moved to. Barash [10] presented the analogy between mean shift clustering and quantum clustering and suggested the capability of using the mean shift procedure for the analysis of gene expression data. From above, we can see that 2D mean shift has currently spread to various areas of science technology, in which tracking ability is the focus.

Mean shift has been already used to find the corresponding control points in images for image registration [13]. In this paper, a novel 3D corresponding control points estimation using mean shift is proposed. It is a nontrivial extension of [13] which described 2D corresponding control points estimation. Unlike [13], in which the probability density function used in mean shift is approximated by the histogram without using spatial information, this 3D mean shift algorithm is not a simple extension from 2D to 3D. In 2D mean shift algorithm the update of the trace point is realized by computing the weight of each point in some 2D region and then the weights are

Received June 7, 2012; Revised September 4, 2012; Accepted September 13, 2012

obtained to adjust the old position of the trace point to a new one. However, for tracking the point in 3D data, the same way is not suitable. Since mean shift algorithm is with no spatial information, we cannot simply compute the probability density of the histogram in a unit volume, which will increase the tracking errors. Experiments have demonstrated that this direct extension from 2D mean shift algorithm to 3D has low tracking accuracy.

Instead, we proposea slice-by-slice way.That is computing the probability density of the histogram in each slice of the search region, and then connecting each probability density function sequentially to form a whole and long probability density function. A cylinder instead of ellipsoid is utilized as a search region to improve the tracking result. In order to decrease the impact of noise and improve the interlaced phenomenon, we use Gaussian function to smooth the probability density, which also increasing the tracking accuracy to some extent. Bhattacharyya coefficient is selected as the metric and compared in each iteration to obtain the new location of the trace point. We also get rid of the original round-off method to process the final trace point and propose creatively three methods to revise the final trace point. Experiments have validated the feasibility and effectiveness of our 3D mean shift algorithm.

2. Research Method

2.1. Mean Shift Algorithms

In this section we first review the mean shift algorithm generalized by Yizong Cheng [3], and then introduce a Bhattacharyya coefficient based metric for target localization proposed by Comaniciu et al. [6].

Let $S \subset X$ be a finite set ("sample"). Let K be a kernel and $w: S \to (0,\infty)$ a weight function. The sample mean with kernel K at $x \in X$ is defined as

$$m(x) = \frac{\sum_{s \in S} K(s-x)\, w(s)\, s}{\sum_{s \in S} K(s-x)\, w(s)} \tag{1}$$

The definition above is the generation of the original one. The difference $m(x)-x$ is called mean shift in Fukunaga and Hostetler [2]. The repeated movement of data points to the sample means is called the mean shift algorithm [2], [14]. This generation improves the original algorithm in the following three ways: 1. Nonflat kernels are allowed. 2. Points in data can be weighted. 3. Shift can be performed on any subset of X , while the data set S stays the same. This generation broadens the range of applications of mean shift, which making the algorithm more practical. More details in [3].

A Bhattacharyya coefficient based metric for target location is proposed by Comaniciu et al., which connecting Battacharyya coefficient and mean shift algorithm. In this integration, Battacharyya coefficient is used to test the similarity of two models. The target is fixed, while the location of the candidate is modified over and over in iterations until the Battacharyya coefficient of the two models is not bigger than the last one. And then we take the last candidate with the biggest Bhattacharyya coefficient as the final corresponding location of the target. By now the correspondence of control points in two models are found. More details in [6].

2.2. 3D Corresponding Control Points Estimation

2.2.1. 3D Mean Shift Algorithm

A novel 3D mean shift algorithm is proposed in this paper, which is based on the 2D mean shift method described in [6] and similarly uses the Bhattacharyya coefficients as a metric. Computing directly the probability density function in the search region is not adapted, instead the method suggests a slice-by-slice way that computes the probability density function slice by slice in the search volume and connects in order the probability density functions of all slices into a whole density vector, using which to calculate weights of points and finally get the new location of the trace point. The slice-by-slice model is illustrated in Figure 1. In this model, the horizontal slices with the same size in both volumes are corresponding. The target model

centered by y_0 is fixed while the target candidate is floating to find the most similar location with the target using our 3D mean shift algorithm.

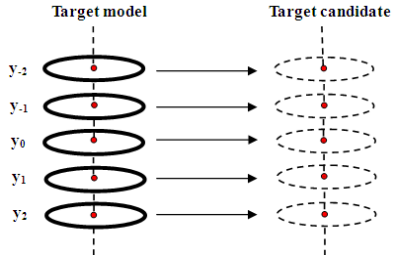

Figure 1. The slice-by-slice model

Now we compute the density function of the target model. Let $\{x_i\}_{i=1\ldots n}$ be the voxel locations of the target model centered at y_0. We define function $b: R^3 \to \{1\ldots m\}$ which associated to the voxel at location x_i the index $b(x_i)$ of the histogram bin corresponding to the grey value of that voxel. The diameter of search region h is $[hx, hy, hz]$. The probability of the grey value u on slice j in the target model is derived by employing a convex and monotonic decreasing kernel profile k which assigns smaller weights to the locations that are farther from the center of the target. Since we adapt slice-by-slice way with cylinder as search region, the center coordinate of each slice is different only at the coordinate z, which is shown in Figure 1. So we assign the center points of different slices as y_j, $j = -\lfloor hz/2 \rfloor \ldots \lceil hz/2 \rceil - 1$ and the points on each slice as $x_{j,i}$. Hence we can represent the density function of the jth slice as

$$\hat{q}_{j,u} = C_{h_j} \sum_{i=1}^{n_{h_j}} k\left(\left\| \frac{y_j - x_{j,i}}{h_j} \right\|^2 \right) \delta\left[b\left(x_{j,i}\right) - u_j \right] \tag{2}$$

Where $j = -\lfloor hz/2 \rfloor \ldots \lceil hz/2 \rceil - 1$, δ is the Kronecher delta function. C_{h_j} is the normalization constant making $\sum_{u=1}^{m} \hat{q}_{j,u} = 1$, $j = -\lfloor hz/2 \rfloor \ldots \lceil hz/2 \rceil - 1$ and the summation of delta functions for $u_j = 1\ldots m$ is equal to one.

Now we introduce vector $\boldsymbol{q}$ to describe the whole density function of all slices in target model, whose size is $j \times u$:

$$\boldsymbol{q} = \bigcup_{j=-\lfloor hz/2 \rfloor}^{\lceil hz/2 \rceil - 1} \hat{q}_{j,u} \tag{3}$$

For the target candidate, the density function is computed as the same way described above, so we can get $\hat{p}_{j,u}$. Also we introduce $\boldsymbol{p}$ as the whole density function of the target candidate.

$$\boldsymbol{p} = \bigcup_{j=-\lfloor hz/2 \rfloor}^{\lceil hz/2 \rceil - 1} \hat{p}_{j,u} \tag{4}$$

It is needed to explain that we define the meaning of $\cup$ as “connect”, shown as Figure 2.

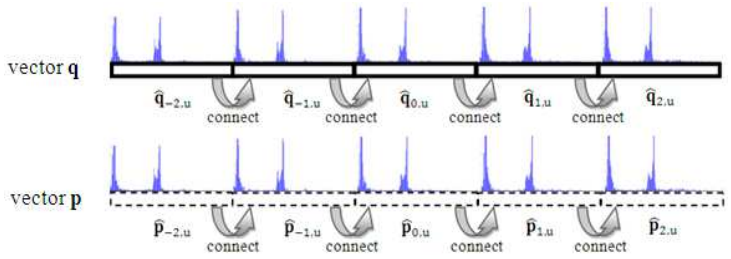

Figure 2. The corresponding probability density vectors

Similar to [6], the new location of the target can be computed as

$$\hat{y}_{new} = \frac{\sum_{j=-\lfloor hz/2 \rfloor}^{\lceil hz/2 \rceil - 1} \sum_{i=1}^{n_h} x_{j,i} w_{j,i} k\left(\left\| \frac{y_0 - x_{j,i}}{h} \right\|^2 \right)}{\sum_{j=-\lfloor hz/2 \rfloor}^{\lceil hz/2 \rceil - 1} \sum_{i=1}^{n_h} w_{j,i} k\left(\left\| \frac{y_0 - x_{j,i}}{h} \right\|^2 \right)} \tag{5}$$

$$w_{j,i} = \sum_{u=1}^{m} \delta\left[b\left(x_{j,i}\right) - u \right] \sqrt{\frac{\hat{q}_{j,u}}{\hat{p}_{j,u}\left(y_j\right)}} \tag{6}$$

We can update the old location in this way mentioned above and then the Bhattacharyya coefficient can be evaluated as:

$$\rho[\boldsymbol{q}, \boldsymbol{p}(y_{new})] = \sum_{j=-\lfloor hz/2 \rfloor}^{\lceil hz/2 \rceil - 1} \sum_{u=1}^{m} \sqrt{\hat{q}_{j,u}\, \hat{p}_{j,u}(y_j)} \tag{7}$$

By now, maximization of Bhattacharyya coefficient can be efficiently achieved based on our mean shift iterations.

2.2.2. Shape of Search Region

In 2D mean shift algorithm we select ellipse centered by trace point as a search region and for 3D mean shift we take it for granted that an ellipsoid centered by trace point should be regarded as a search region. While in practical trials, it's difficult to track precisely in two volume data using a region of ellipsoid. The reason is that the number of points in each slice will decrease as the distance between the slice and the centered slice increases, that is the slice furthest from the center has the least points and the center slice has the most points. In this case, if points move along z direction, although the weights of points in small slice are bigger, the energy of pulling the whole data along z is small due to its fewer points, which making the further slice contribute less to the new location of trace point. In view of that situation, we try to change the shape of search region from ellipsoid to cylinder and experiments show that the algorithm modified has an improved performance in tracking points. The model with ellipsoid as search region is shown in Figure 3.

Then the comparison of tracking results between ellipsoid and cylinder is shown in Table 1. The experimental data used is a 64×64×94 brain volume data, the target candidate is obtained through moving the target one by [2 2 3], that is the trace point in target is [29 38 63] and the same point in target candidate is [31 40 66]. Not a suitable h is found in ellipsoid region to track the right point in plenty of trials and at least one suitable h, [10 8 10], is found in cylinder region to make the tracking correct.

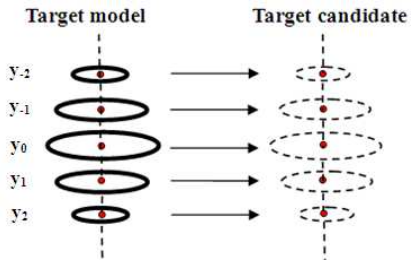

Figure 3. The model with search region of ellipsoid

Table 1. Comparison of search region between ellipsoid and cylinder. The bold item is the optimal result.

	Ellipsoid	cylinder
h	y	y
[4 4 4]	[29 38 63]	[29 40 64]
[4 4 6]	[29 38 62]	[29 40 64]
[6 6 6]	[29 40 64]	[29 41 65]
[8 8 8]	[29 38 63]	[33 39 66]
[8 8 9]	[29 39 63]	[33 40 64]
[10 8 10]	[29 38 63]	**[31 40 66]**

2.2.3. Smoothing the Probability Density Function with Gaussian Function

When computing the probability density function of the target model and target candidate, it's possible that the density of one level exists in one, but is none (zero) in the other, which making weight $w_{j,i}$ computation of points impossible. Taking this situation into account, we try to use Gaussian function to smooth the probability density functions in order to improve this case. A 1×3 Gaussian temple is adapted, which only involving two gray levels around the center. A 1×5 or bigger Gaussian temple is not reasonable because of small impact between non-adjacent levels. The difference before and after the Gaussian smoothing is shown in Figure 4, from which we can see that the interlaced phenomenon can be improved effectively and the computation of weights of points becomes available.

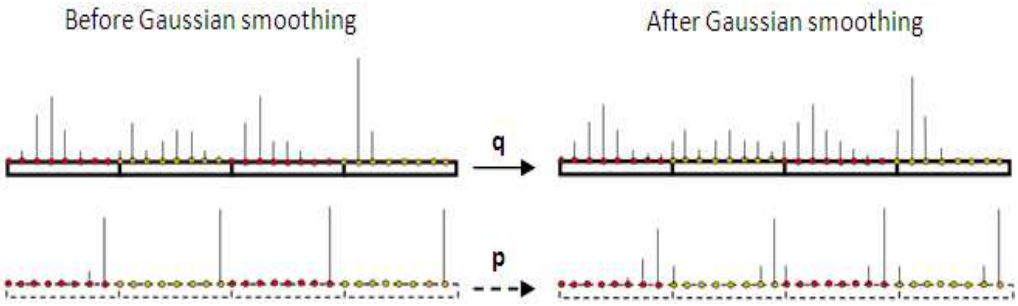

Figure 4. Comparison before and after Gaussian smoothing

2.2.4. Revising the Tracking Locations

It's noTable that the new location computed iteratively through weights of points is not an integer. In the original algorithm of mean shift, the floating point is just rounded off directly, which is called round-off method. Operation of this kind is so simple that big errors will occur after multi-iteration of computing locations of the trace point, which will result in a small Bhattacharyya and quit the iteration in advance. In view of that, we try to revise the location of each iteration in the following three ways according its moving tendency, in which the so-called moving tendency is making the computed location reach its possible bottom or top position, that

is in each iteration if the old location ($location_{old}$) moves along the positive direction and the floating moving distance is $offset$, then we can define the new location of this iteration as $\lceil location_{old} + offset \rceil$. Similarly if the old location moves along the negative direction, the new location we define is $\lfloor location_{old} - offset \rfloor$. The possible revising results of trace points with different methods are shown in Figure 5 and the three revising methods are in details below. The three revising methods are integrated in the mean shift algorithm and calculated in each mean shift iteration.

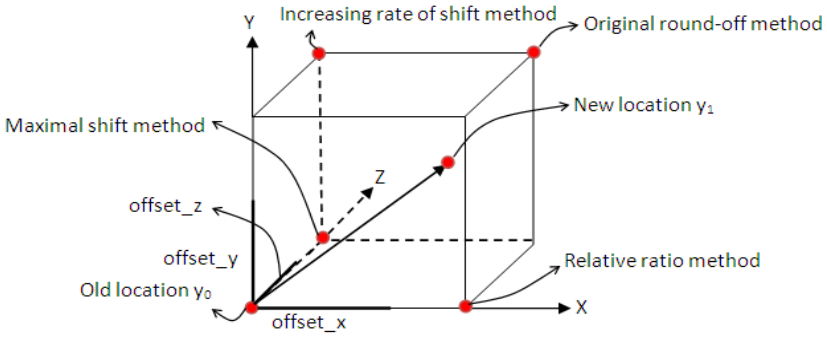

Figure 5. Comparison of three revising ways with different possible locations

Maximal Shift Method. This method revises the floating location in terms of the max shift of one coordinate. It is described below.

(i) Compute the shifts along x, y, z direction between the old and the new locations. $\text{cor\_offset} = [\text{x\_offset}, \text{y\_offset}, \text{z\_offset}]$;

(ii) Initialize y_2 and let $y_2 = y_0$; (y_0 is the old location and is an integer.)

(iii) Find the max one in cor_offset and revise its corresponding coordinate according to the tendency of movement. The revised location is assigned to y_2 .

(iv) Let $y_1 = \lfloor y_1 + 0.5 \rfloor$ (y_1 is the new location computed and is a floating point.). Compute respectively the probability density functions of y_1 and y_2 ; compare their Bhattacharyya coefficients and assign the location with the max Bhattacharyya coefficient to the finally new location.

Relative Ratio Method. This method revises the floating location according to the relative ratio among the shifts of x, y, z . Below is the procedure of this method.

(i) Compute the shifts along x, y, z direction between the old and the new locations. $\text{cor\_offset} = [\text{x\_offset}, \text{y\_offset}, \text{z\_offset}]$, $\text{cor\_abs} = |\text{cor\_offset}|$;

(ii) Sort the cor_abs in descending order: $\text{cor\_sort} = [\max, \text{medium}, \min]$;

(iii) Obtain the relative ratio: $\text{cor\_ratio} = [\text{medium} / \max, \min / \max]$;

(iv) Define adaptively thresh: $\text{thresh} = ((\max + \text{medium} + \min) / 3) / \max$;

(v) Count the number that makes elements of cor_ratio bigger than thresh .

If number is 0, revise only the $\max$ corresponding coordinate according to its moving tendency.

If number is 1, revise the $\max$ and medium corresponding coordinates according to their moving tendency.

If number is 2, revise the three coordinates according to their moving tendency.

Increasing Rate of Shift Method. This method revises the floating location using the increasing rate of shift. Details are below.

(i) Compute the shifts along x, y, z direction between the old and the new locations. $\text{cor\_offset} = [\text{x\_offset}, \text{y\_offset}, \text{z\_offset}]$ $\text{cor\_abs} = |\text{cor\_offset}|$;

(ii) If it's the first iteration, define $\text{cor\_base} = \text{cor\_offset}./\text{cor\_abs}$ as the initial base;

(iii) Define the increasing rate of shift: $\text{cor\_rate} = \text{cor\_offset}./(\text{cor\_base} + \varepsilon)$;

(iv) Find the points with the rate of increase less than zero and assign their rates of increase by zero. That is points of this kind don't participate in the following comparison. Since a negative rate of increase means the point moving oppositely in two continuous iterations, which we think is meaningless and unfair to compare the rate of increase.

(v) Define two variables thresh_offset and thresh_base. The reason to introduce those two variables is that if cor_offset(i) and cor_base(i) are both very small, which means those points are actually not the ones we want, while the quotient of two small values may be big, that will make the rate of increase of this kind of points big. In order to make our algorithm robust, we should eliminate those points by selecting variables thresh_offset and thresh_base by hand to constrain this situation.
Find the points that making $|\text{cor\_offset(i)}| < \text{thresh\_offset}$ or $|\text{cor\_base(i)}| < \text{thresh\_base}$, where $\text{i} = 1...3$. Assign the corresponding increasing rate of those points by zero.

(vi) Define adaptively the thresh. Let $\text{thresh} = \frac{1}{3}\sum_{i=1}^{3} \text{cor\_rate(i)}$.

(vii) Define and initialize the direction vector: $\text{direction} = [0, 0, 0]$; the direction vector is used to label the moving direction of landmarks, "1" means the points moving along positive direction while "-1" means the points moving along opposite direction, it mainly aims at modifying cor_base. Once a point is revised, the just used increasing rate is no longer useful, that means we should reset cor_base with "1" or "-1" on the corresponding position. So the vector direction is just here to remark the moving direction of revised points and later we use it to reset the cor_base.

(viii) Initialize $y_1 = y_0$;

(ix) Find the coordinate making the rate of increase bigger than thresh and revise it according to its moving tendency.
When revising it, If the value of coordinate increases, let $\text{direction(i)} = 1$; If the value decreases, let $\text{direction(i)} = -1$; Otherwise, keep direction(i) unchanged.

(x) Find the changed one in direction vector and modify the cor_base in the same position with the changed value in direction. And then go to the next iteration.

3. Results and Analysis

3.1. Experimental Data

Experimental data used here is the same with [11], [12] provided by Dir-lab. From the website we can get ten case data which represent ten patients. Each case includes the set of 10 component phase CT images from T00 to T90. All images are 16-bit integer class. Also included with each case are corresponding landmarks marked manually. These corresponding landmarks can be used to validate our algorithm, where the landmarks in the T00 image are source control points, and our method is used to estimate the corresponding points in other phase images. The more coincidence between tracked points with our method and the ones picked by hand from Dir-lab, the better our algorithm. Considering the corresponding points positioned by hand, in which position error is eviTable, it makes our tracked position with minor deviations accepTable.

We process properly the data provided above before our experiment, since the data mentioned is 16-bit integer, the histogram of the data shows a top-heavy phenomenon. That is the peak occurs in small value and the end part of histogram is sparse. In view of that, we choose a suitable gray range of the data that can describe the content as fully as possible and map the 16-bit data to 8-bit data. For the problem that one landmark in different time point data may be different in gray value, we take that situation into account to choose a gray range making the difference as little as possible and finally choose the gray range of 0-900 to process the case1. All the experiments in this paper are based on the choice described above.

3.2. Comparison between Comaniciu's Way and Slice-by-slice Way

To compare the difference of probability density estimations between our method in this paper and Comaniciu's, we use the data case1_T00_s and case1_T50_s described above to test and compare the two different ways. The T00 data is fixed as the target, while T50 data is the target candidate. In order to compare them fairly, no revising method is utilized here, that is both methods adapt the direct round-off way to the new locations. To estimate the 3D corresponding control points, we introduce Total error and Count, respectively representing the Euclidean distance error and the total number of precisely tracked points. Average is the average error of 75 points, Average = total error / 75.

The comparison results between Comaniciu's way and our slice-by-slice way is shown in Table 2. Data used here are case1_T00_s and case1_T50_s described in details in Dir-lab, and the trace point sets are 75 corresponding landmark pairs of two time points mentioned above. From it, we can see that the tracking performance is distinctly improved. From Comaniciu's way to our slice-by-slice way, total error decreases by more than 20, and count increased by 7.

Table 2. Tracking result comparison between Comaniciu's way and slice-by-slice way.

	Total error	Count	Average
Comaniciu's way	134.097	2	1.7880
Slice-by-slice	112.407	9	1.4987

3.3. Gaussian Smoothing Comparison and Results of Three Revising Ways Compared with the Round-off Method

Next, we compare the difference with and without Gaussian function to smooth the probability density. And then three statistics of tracking errors are used in our experiments, including the general tracking statistic, Euclidean distance error statistics and Distance statistics on each coordinate. They all show the same results, but from different views. The tracking result parts represent the general situation of tracking including Count, Total error and Average described above. All the experiments are tested in the condition that h = [7 7 8], m=8 with the volume data case1_T00_s and case1_T50_s and the corresponding control points in those two volume data are respectively saved in arrays corT00 and corT50. The coordinates of our tracking results are saved in corfinal.

The difference before and after Gaussian smoothing is shown in Table 3. The three revising methods are all better than the rounded off method. Before Gaussian smoothing, the maximal shift method reaches the best result with minimal total error 100.4205, minimal Average 1.3390 and maximal count 13. And the second best is increasing rate of shift method. Relative ratio method is not as good as other two ways but is better than the round-off way. We can also see that the tracking performance is improved in all methods after Gaussian smoothing, in which the maximal shift method is also the best with total error 99.4715.

Table 3. The tracking results before and after Gaussian smoothing using different methods. Thethree revising methods are all better than the round-off method. And after Gaussian smoothing, the tracking ability of all methods are improved.

	No Gaussian smoothing			Gaussian smoothing		
	Count	Total error	Average	Count	Total error	Average
Round-off method	9	112.4057	1.4987	10	109.6867	1.4625
Maximal shift method	13	100.4205	1.3390	14	99.4715	1.3263
Relative ratio method	13	103.4173	1.3789	15	101.6537	1.3554
Increasing rate of shift method	13	100.5809	1.3411	15	99.9453	1.3326

The Euclidean distance error statistics with Gaussian smoothing between corfinal and corT50 is shown in Figure 6, in which the suitable sigma are selected respectively for different methods and the horizontal axis represents the distribution of different error distances while the vertical axis the number of points corresponding to the error distances they have. To the tracking results, the more the front parts occupy, the better the methods. It is shown that the

number of error distance distributions from 0 to 2 is both 63 (13+25+25=63) for the maximal shift method and the increasing rate of shit method. That means the two methods are both with good performance in some degree. The relative ratio method is not as good as two other ways, but much better than the round-off one.

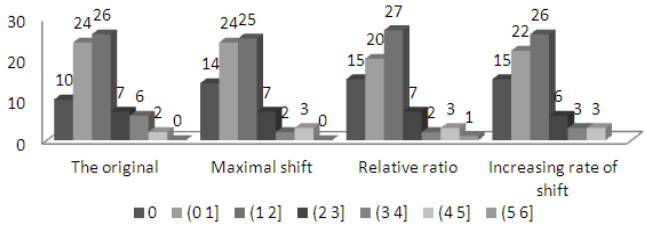

Figure 6. Euclidean distance error statistics with Gaussian smoothing between corfinal and corT50. The Maximal and Increasing rate methods both present good performance in tracking results after Gaussian smoothing.

We choose two best results from plenty of trials using increasing rate of shit method. Table 4 shows the optimal tracking results, which making this method the best with the minimal total error and the most exact tracked points.

Table 4.Optimal tracking results comparison with different thresh_offset and thresh_base pairs.

Thresh_offset	Thresh_base	count	Total error	Average
0.1	0.25	14	**98.1168**	1.3082
0.2	0.25	15	**98.4427**	1.3126

The distance error statistics on each coordinate is shown in Table 5, that is on x, y, z coordinate respectively. The first row is the error distance distribution from 0 to 4, in which the error distance is calculated in pixels; the first column lists four different methods. Since the number of trace points is 75, so the number of total coordinates of those points is 225 (75×3=225). From the Table, we can see that the increasing rate of shift method can reach the best with 122 exact tracked points, 86 points of one-pixel distance error and 11 points of two-pixel distance error. The second best is the maximal shift method.

Table 5. Distance error statistics on each coordinate between corfinal and corT50.

	0	1	2	3	4
The original method	110	93	14	5	3
Maximal shift method	121	86	12	3	3
Relative ratio method	119	88	11	3	4
Increasing rate of shift method	122	86	11	4	2

We select several control points to track and Figure 7 illustrates the tracking results in slices. Each row stands for a phase and corfinal row is our tracking results. Each column has the same trace point with different coordinates in different phases. Coordinates in corT00 are initial locations of points; coordinates in corT50 are the exact corresponding locations of corT00 and the ones in corfinal are our tracking results of corresponding landmarks. Tracking regions are marked by green circles.

Finally, a 3D image of tracking results is illustrated in Figure 8, in which we compare the Comaniciu's way and our method (our method includes all the ideas mentioned above and finally uses the increasing rate of shift method as a revising method) vividly. In this Figure, red points are the final points we actually tracked using two different algorithms respectively. Blue points are the corresponding exact points provided by Dir-lab. Green segments are the Euclidean distance describing the tracking errors of two different methods.

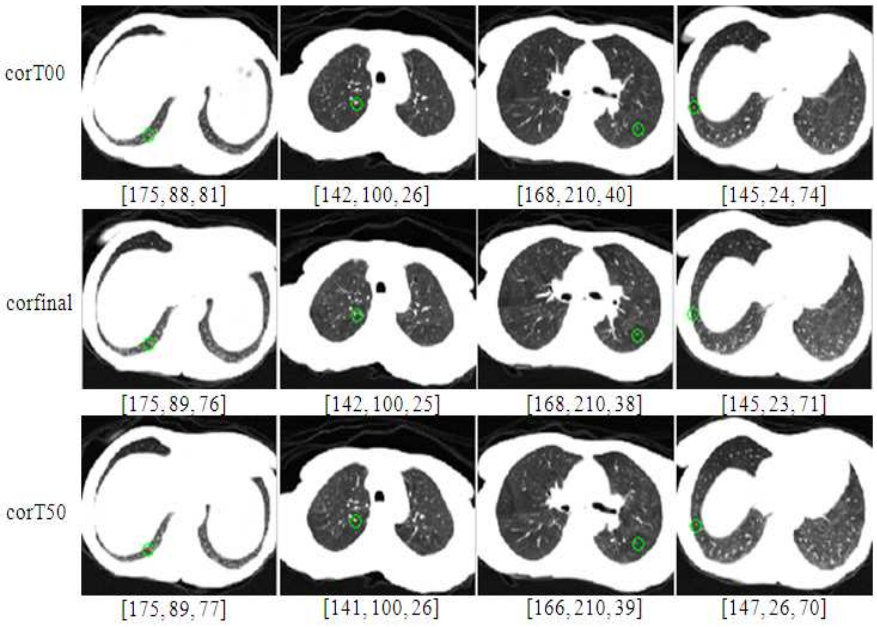

Figure 7. Illustration of tracking results.

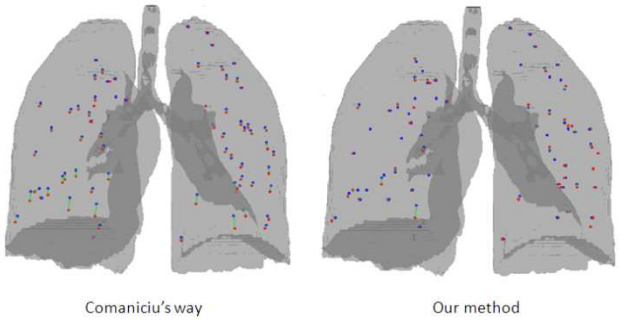

Figure 8. Comparison of tracking results between two methods in 3D images. The actual tracked positions using two methods are labeled with red points; the corresponding exact positions of trace points are labeled in blue; the tracking error distances are labeled with green segments.

Experiments show that the slice-by-slice idea of 3D mean shift algorithm proposed in this paper can make the point tracking in volume available and the tracking effect is much better than the original way. It also indicates that a cylinder search region is more suitable for our algorithm. The connection of the two ideas improves the tracking ability of point in 3D data. In addition, smoothing the probability density function with Gaussian function can not only resolve the problem of interlaced phenomenon, but also raise the tracking accuracy in some degree. Meanwhile, for data used in our experiments, increasing rate of shift method used to revise the location reaches the minimum tracking error with 98.1168, next is the maximal shift method with 99.4715. The relative ratio method with 101.6537 is not as good as the other two ways mentioned above, but is better than the round-off method. It's remarkable that the relative ratio method adapts the way through selecting threshes automatically. The reason is that selecting thresh by hand will result in a situation that smaller thresh will make all the coordinates changed, while bigger thresh will make this method similar to the maximal shift method. So we try to make the method select thresh automatically, that is computing thresh in terms of situation of each point in each iteration to revise the location, and that sounds reasonable.

4. Conclusion

3D mean shift algorithm proposed in this paper is not a direct extension of 2D mean shift and the reason is that the complexity of 3D spatial information increases compared with 2D images. So the original algorithm cannot meet the demand of high-accuracy in 3D point tracking, based on which a novel slice-by-slice idea is proposed. This algorithm can effectively avoid the limit of original mean shift algorithm without spatial information. Three new revising methods also contribute to the tracking results. From the experiments, this algorithm can track point in 3D data effectively which plays an important role in feature point-based volume registration. So the algorithm in this paper is valuable in some sense.

Acknowledgments

The authors acknowledge the National Natural Science Foundation of China for providing funds for this research work (No. 60972112).

References

[1] Goshtasby A. 2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications. Wiley Press. 2005.
[2] Fukunaga K, Hostetler LD. The Estimation of the Gradient of a Density Function with Applications in Pattern Recognition. *IEEE Trans. Information Theory*. 1975; 21:32-40.
[3] Cheng Y. Mean Shift, Mode Seeking, and Clustering. *IEEE Trans. Pattern Analysis and Machine Intelligence*. 1995; 17(8): 790-799.
[4] Comaniciu D, Meer P. Mean shift: A Robust Approach toward Feature Space Analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*. 2002; 2(5):603-619.
[5] Comaniciu D, Meer P.*Mean Shift Analysis and Applications*. Proceedings of the Seventh Intl. Conf. Computer Vision. 1999: 1197-1203.
[6] Comaniciu D, Ramesh V, Meer P. Real-Time Tracking of Non-Rigid Objects using Mean Shift. *IEEE Computer Vision and Pattern Recognition*. 2000; 2:142-149.
[7] Collins R. *Mean-shift Blob Tracking through Scale Space*. IEEE Conf. Computer Vision Pattern Recognition. 2003; 11: 234-240.
[8] Avidan S. *Ensemble Tracking*. IEEE Conf. on Computer Vision and Pattern Recognition. Sandiego. 2005; 2: 494-501.
[9] Comanciu D, Visvanathan R, Meer P. Kernel-Based Object Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*. 2003; 25(5): 564-575.
[10] Barash D and Comaniciu D. *Meanshift Clustering for DNA Microarray Analysis*. IEEE Computational Systems Bioinformatics Conference(CSB). 2004: 578-579.
[11] Castillo R, Castillo E, Guerra R, Johnson VE, McPhail T, Garg AK, Guerrero T. A Framework for Evaluation of Deformable Image Registration Spatial Accuracy using Large Landmark Point Sets. *Phys Med Biol*. 2009; 54: 1849-1870.
[12] Castillo E, Castillo R, Martinez J, Shenoy M, Guerrero T. Four-dimensional Deformable Image Registration using Trajectory Modeling. *Phys. Med. Biol*. 2009; 55: 305-327.
[13] Yang X, Pei JH. Rapid and Robust Medical Image Elastic Registration using Mean Shift. *Chinese Optics Letters*. 2008; 6(12):950.
[14] Silverman BW. Density Estimation for Statistics and Data Analysis. London: Chapman and Hall. 1986.
[15] Fahmi Fardiyan Arief, Muchlas, Tole Sutikno. Kompas Digital Dengan Output Suara Berbasis Mikrokontroler AT89S52. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2008; 6(1):1-6.
[16] Sarwosri, Darlis Herumurti, Indri Sulistyowati. Klasifikasi Secara Efisien pada Database Multi Relasi dengan Algoritma Crossmine. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2008; 6(1):7-14.