2007－CVIM－159 (2)
2007／5／14

K-means Clustering Based Pixel-wise Object Tracking

CHUNSHENG HUA,[†,☆] HAIYUAN WU ,[†] QIAN CHEN [†]
and TOSHIKAZU WADA [†]

This paper brings out a robust pixel-wise object tracking algorithm which is based on the K-means clustering. In this paper, the target object is assumed to be non-rigid and may contain apertures. In order to achieve the robust object tracking against such objects, several ideas are applied in this work: 1) Pixel-wise clustering algorithm is applied for tracking the non-rigid object and removing the mixed background pixels from the search area; 2) Embedding the negative samples into K-means clustering so as to achieve the adaptive pixel classification without the fixed threshold; 3) Representing the image feature with a color-position feature vector so that this algorithm can follow the changes of target colors and position simultaneously; 4) A variable ellipse model is used to restrict the search area and represent the surrounding background samples; 5) Tracking failure detection and recovery processes are brought out according to both the target and background samples; 6) A radial sampling method is brought out not only for speeding up the clustering process but also improving the robustness of this algorithm. We have set up a video-rate object tracking system with the proposed algorithm. Through extensive experiments, the effectiveness and advantages of this K-means clustering based tracking algorithm are confirmed.

1. Introduction

Years of vision research have yielded in lots of powerful object tracking algorithms. Numerous powerful and effective algorithms have been applied into object tracking, such as the Kalman filter [7], template matching [8], EM algorithm [11], CONDENSATION [9] (also called as the "particle filter") algorithm, dynamic Bayesian network [10], mean shift [12] and iterative clustering [6].

In [16], the original template matching algorithm uses the sum of grey value of the pre-defined template to describe the target feature. The object tracking is achieved by searching over the image to find out the region that has the maximum similarity to the pre-defined template. Because the grey value of template is used to describe the target feature, the original template matching is very sensitive to the illumination changes and target deformation. Improved works on template matching [8] have been reported.

Background subtraction is able to track the target object by checking the differences between the observed image and the pre-defined background model. Therefore, it is especially suitable to work indoors, such as the subway station, airport, etc. Although adaptive background subtraction [17] has been proposed, background subtraction still suffers from the random noise (such as rains or snows) and is not suitable to work in the dynamic scene. The same problem also exists in the optical flow method [18] which tracks the target object by examining the characteristics of flow vectors of the moving target over the surrounding background.

In [12], Comaniciu *et al.* use the color distribution of the target to describe the target feature and bring out the mean-shift tracking algorithm, which can achieve the robust object tracking for the non-rigid objects under the cluttered condition. One of the key issues in mean shift algorithm is to produce the center-weighted image (which is also a 3D color histogram) at time with the Bhattacharyya coefficient. In this center-weighted image, the pixels on the target object have high weight, while pixels on the background have low weight. By applying the hill-climbing calculation to find out the peak within such weight image, mean shift algorithm achieves the robust object tracking under cluttered condition. But means shift algorithm is not suitable for the monochromatic and planar objects, because such objects usually appear like a vertical line or a narrow peak in the color histogram; even the illumination just changes a little, such narrow peak will be drifted dramatically and hill-climbing method will fail because there is no overlap of the target template in two adjacent frames.

CONDENSATION [9] can track the target through occlusion and clutter by reasoning over a state-space of multiple hypothesis. Because it combines the random sampling techniques with the posterior probability of the target object together, it achieves very robust object tracking.

However, although these algorithms are quite powerful, they only concentrate on the similarity between the target model and an unknown re-

† Faculty of System Engineering, Wakayama University
☆ Presently with the ISIR of Osaka University

gion/pixel. In order to measure such similarity, a threshold is usually applied into these algorithms. Since the target object may move under the cluttered condition, it is difficult to select the proper threshold to work stably under all conditions. Furthermore, there is no guarantee that if the object with the maximum similarity is really the target one or not.

Collins [14] *et al.* bring out an idea that: while object tracking, the most important thing is the ability to discriminate the target object from its surrounding background. They propose a method to switch the mean-shift tracking algorithm among the different linear combination of the *RGB* colors which can select the very features that distinguish the object most from the surrounding background. Therefore, the kernel of this paper is how to select the best combination of *RGB* colors to discriminate the target from background. The combination that achieves the maximum ratio between the color distribution of target and that of the background is considered as that best features. Improved performance compared with the standard mean-shift algorithm has been reported in that paper. Even so, the color histogram has very little identification power and this method appears to work only when the target is solid and its appearance does not change dramatically. Meanwhile, in the case of high dimensional features like textures, the large number of combination of colors (in fact, each image contains 49 such color combinations) will prevent it from achieving the real-time performance.

Similar ideas have been applied by Nguyen [15] and Zhang [19]. In [19], they describe the target and background features with their color histograms. An unknown pixel is classified into target or background by comparing the bin values of its color between the target histogram and the background one. Therefore, when some parts of the surrounding background contains similar color to that of target, this ratio will become very low and wrong conclusion will be made. Furthermore, because they assume that both the target and background are solid, the long-term performance of this paper is not ensured. Meanwhile, this work can not be applied to track the object with apertures because the mixed background will pollute the target histogram. In [15], although the more powerful Gabor filter is used to discriminate the target from the background, the performance of that work in the long term is suspected because the target is assumed to be solid. When the target is non-rigid, there will be no guarantee that the update of target template is correct, and the multi-scale problem is also remained in [15].

Therefore, obviously almost all the mentioned tracking algorithms share one common problem: when the target object is non-rigid and/or contains apertures, background pixels will be mixed into the target object; when the target object moves under the cluttered background condition, the continuously mixed background pixels will greatly degrade the purity of the target feature (such as color or texture). In this paper, this phenomenon is called as *background interfusion.*

In order to solve this background interfusion problem, we bring out a pixel-wise object tracking algorithm which is based on applying a reliability-based K-means clustering algorithm into both the target and background samples. This reliability-based K-means clustering algorithm is applied within a variable ellipse model which is used to restrict the search area as well as represent the representative surrounding background samples. According to the triangular relationship among an unknown pixel, its nearest target center and its nearest representative background center, each pixel will be assigned with a reliability value which indicates how reliable it is to classify this pixel into target or background groups. Noise pixels that neither belong to the target clusters nor the background clusters will be given low reliability value. Then, under such reliability, the probability that this pixel belongs to the target /or background clusters will be computed. Embedding this probability into K-means clustering, the proposed algorithm is able to remove the background pixels as well as the noise pixels from the target object. By representing the image feature with a color-position feature vector and updating the target centers with such feature vector, this tracking algorithm can smoothly follow the position and color changes simultaneously. Since both the target centers and the background ones are continuously updated while object tracking, this algorithm achieves the adaptive object tracking without threshold. A radial sampling method is used not only for speeding up but also improving the robustness of the proposed algorithm. Finally, with both the target samples and background samples, this algorithm achieves the automatic tracking failure detection and recovery.

2. K-means Clustering Based Object Tracking

2.1 Image Feature Representation

In order to describe the image feature properly, all the objects (including the target object and the background objects) located in the image are rep-

resented by the following assumption: each object is composed of one or several regions, all the pixels within each region should contain two properties — 1) all the pixels located within one region should contain similar colors to each other — 2)the pixels within each small region should be close to each other in the 2D image space.

Therefore, according to this assumption, the image feature is described by a constructed color-position 5D feature vector "$\mathbf{f} = (\mathbf{c}, \mathbf{p})^T$". Any kind of color system can be applied to this feature vector (such as the "*RGB*", "*CMY*", "*YIQ*", "*YCbCr*" and "*YUV*" color system). Here, "$\mathbf{c} = (Y, U, V)^T$" is used to describe the color and the position of a pixel is described by "$\mathbf{p} = (x, y)^T$". That is because the output of the camera applied in this tracking algorithm is the "*YUV*" colors and the directional application of "*YUV*" colors will reduce the processing time.

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & 0 & \alpha \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \\ x \\ y \end{bmatrix}, \quad (1)$$

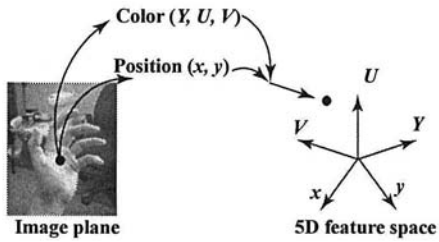


Fig. 1 Explanation for the 5D feature vector.

In this 5D feature space, the Euclidean distance between two points $\mathbf{f}_1$ and $\mathbf{f}_2$ is calculated as:

$$\| \mathbf{f}_1 - \mathbf{f}_2 \|^2 = \| \mathbf{c}_1 - \mathbf{c}_2 \|^2 + \alpha \| \mathbf{p}_1 - \mathbf{p}_2 \|^2 . \quad (2)$$

By performing the pixel classification in such a feature space, the proposed tracking algorithm has the ability to simultaneously follow the changes of position and color of the target.

2.2 Pixel-wise Object Tracking with K-means Clustering

2.2.1 K-means Clustering with Negative Samples

In this paper, since the target object is allowed to be non-rigid and contain apertures within it, the conditional model-based object tracking algorithms are unsuitable for such object, because they usually assume the target object to be priorly known and solid.

In order to solve the problems of the model-based tracking algorithms, we select the K-means clustering algorithm for object tracking, because it: 1) Does not need any prior target appearance model, so the initialization becomes simple; 2) Can track the non-rigid (human body) and object with aperture (hand) well, even the target object deforms greatly.

However, most of the conventional pixel-wise tracking algorithms only use the target information for pixel classification, which is similar to those model-based methods. Thus in those works, they only measure the similarity (or dissimilarity) between an unknown pixel and the target sample. Therefore, a threshold is required to determine whether this pixel belongs to the target group or not, according to the estimated similarity. However, it is well known that, while object tracking, both the target appearance and the surrounding background can change easily due to the illumination variance, target deformation, etc, so a fixed threshold can not be suitable under different conditions.

In this paper, we introduce the concept of non-target and apply the K-means clustering to both the target and non-target samples. The motivation to introduce this concept is that: through our experiments, we found out that the ability to discriminate the target from its surrounding background is the key element that resolves the tracking success or failure. Therefore, it is reasonable to consider that a good tracking algorithm should be performed between the target and background information. In this work, we introduce the concept of non-target (background) for the target discrimination. As shown in **Fig.**2, whether an unknown pixel within the searching area is the target or not is discriminated by measuring its similarity to the target group and non-target ones.

In this case, the similarity measurement becomes a problem of "*yes*" or "*no*" and we consider this measurement is better than only comparing the similarity with the target. That is because, with the negative reference, it will be more confirmable that the pixels that do not belong to the non-target groups will belong to the target group. In **Fig.**2, *RTB* means the "*Real Target Boundary*" that really discriminate the target from the background. Since it is difficult to stably get the *RTB*, we propose a model named as "*absolute target region*" (hereafter *ATR*) to approximate *RTB* and describe the target object and its surrounding background. The *ATR* is an area that contains all the target object pixels,

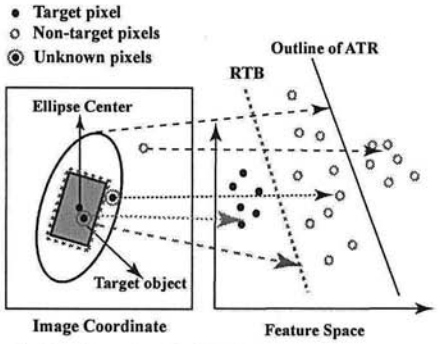


Fig. 2 Illustration for describing non-target samples

meanwhile, all the pixels on the outline of *ATR* are the representative non-target pixels around the target object.

In this paper, we use the *ATR* to: 1) restrict the search area for speed up and increasing the robustness of our tracking algorithm; 2) represent the approximate target shape and direction; 3) model the surrounding non-target background. Since both the target sample and surrounding *ATR* are continuously updated, our method becomes robust against the target appearance variance.

2.2.2 Reliability-based K-means Clustering

However, two problems degrade the performance of K-means clustering algorithm: 1) When the noise data not belonging to any cluster exist, K-means clustering will wrongly classify them into some pre-defined clusters; 2) the wrongly assumed number of clusters sometimes leads to the wrong clustering result. Although some researchers brought out some improvements [1–5] on K-means clustering, while object tracking such problems still affect the performance of K-means clustering.

(a) Input image (b) Result of K-means

Fig. 3 In [13], K-means tracker will suffer from the noise data.

The problem caused by the noise data can be explained as: the K-means clustering only classifies the unknown data according to its distance to the prototype (or cluster centers). As shown in **Fig.**5, K-means clustering algorithm only check d_i and d_j (which are the distance among an unknown data $\mathbf{x}$ and the cluster centers $\mathbf{m}_i, \mathbf{m}_j$) to judge which cluster $\mathbf{x}$ should belong to, even if $\mathbf{x}$ is more like to be a noise data.

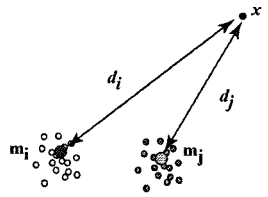


Fig. 4 The illustration of EKM, when the noise data exists.

In order to solve the mentioned problems, we introduce the reliability estimation into K-means clustering. The *reliability* value for each data is used to judge if it is reasonable to classify an unknown data into one cluster or not.

Because the noise data are always distant from the data belonging to the initialized clusters (called as normal data), the distance from noise data to its nearest cluster center should be longer than that of the normal data. Therefore, the distance between a data and its closest cluster center can be used to classify the unknown data. However, a standard is necessary for estimating how far a data is from its nearest cluster center. Here, we use the triangular relationship among an unknown data and its two nearest cluster centers to measure if it is reliable or not to classify this data into some clusters. A high reliability value should be assigned to the normal data, and a low value to the noise one.

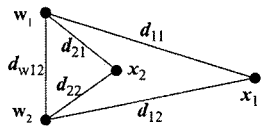


Fig. 5 The relationship between data vectors and their two closest cluster centers.

As shown in **Fig.**5, $\mathbf{w}_1$ and $\mathbf{w}_2$ are the cluster centers, $\mathbf{x}_1$ and $\mathbf{x}_2$ are the unknown data vectors. Only according to d_{11} and d_{21}, we can not judge $\mathbf{x}_1$ and $\mathbf{x}_2$ to be the noise data vectors or the ones that belong to some clusters except that $\mathbf{x}_2$ is closer to $\mathbf{w}_1$ than $\mathbf{x}_1$. However, according to the shape of triangle $\triangle\mathbf{x}_1\mathbf{w}_1\mathbf{w}_2$ and $\triangle\mathbf{x}_2\mathbf{w}_1\mathbf{w}_2$, we can draw a conclusion that, compared with $\mathbf{x}_2$, $\mathbf{x}_1$ has higher probability to be a noise data vector.

While clustering an arbitrary data vector $\mathbf{x}_k$ of data set $\mathbf{X}$ ($\{\mathbf{x}_k; k = 1, \ldots, n\}$), the reliability value of $\mathbf{x}_k$ is defined as the ratio of the distance between its two closest cluster centers to the sum of the dis-

tance from $\mathbf{x}_k$ to the two cluster centers:

$$R(\mathbf{x}_k) = \frac{\| \mathbf{w}_{f(\mathbf{x}_k)} - \mathbf{w}_{s(\mathbf{x}_k)} \|}{d_{kf} + d_{ks}}, \tag{3}$$

where

$$\begin{aligned} d_{kf} &= \| \mathbf{x}_k - \mathbf{w}_{f(\mathbf{x}_k)} \|, \\ d_{ks} &= \| \mathbf{x}_k - \mathbf{w}_{s(\mathbf{x}_k)} \|. \end{aligned} \tag{4}$$

$f(\mathbf{x}_k)$ and $s(\mathbf{x}_k)$ are the subscript of the closest and the secondly closest cluster centers to $\mathbf{x}_k$:

$$\begin{aligned} f(\mathbf{x}_k) &= \underset{i=1,\ldots,c}{\operatorname{argmin}} (\| \mathbf{x}_k - \mathbf{w}_i \|) \\ s(\mathbf{x}_k) &= \underset{i=1,\ldots,c, i \neq f(\mathbf{x}_k)}{\operatorname{argmin}} (\| \mathbf{x}_k - \mathbf{w}_i \|). \end{aligned} \tag{5}$$

The degree (μ_{kf}) of a data vector $\mathbf{x}_k$ belonging to its closest cluster $f(\mathbf{x}_k)$ is computed from d_{kf} and d_{ks}:

$$\mu_{kf} = \frac{d_{ks}}{d_{kf} + d_{ks}}. \tag{6}$$

Since $R(\mathbf{x}_k)$ indicates how reliable that $\mathbf{x}_k$ can be classified, the probability that $\mathbf{x}_k$ belongs to its closest cluster can be computed as the product of $R(\mathbf{x}_k)$ and μ_{kf}.

$$t_{kf} = R(\mathbf{x}_k) * \mu_{kf} \tag{7}$$

t_{kf} denotes that: under the reliability $R(\mathbf{x}_k)$, the probability that $\mathbf{x}_k$ belongs to its closest cluster.

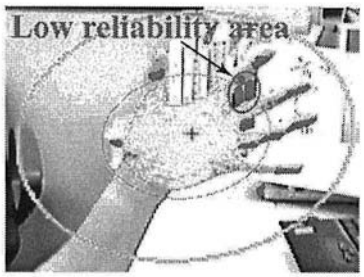


Fig. 6 RK-means clustering can detect the outliers with the reliability evaluation.

2.2.3 Data grouping

Assuming the number of clusters and the initial cluster centers are given, the data grouping in RK-means clustering algorithm is carried out by two steps:

1. For each data vector $\mathbf{x}_k$, compute its probability of belonging to its nearest cluster center as Eq(7).
2. Update the clusters by minimizing the following objective function:

$$J_{rkm}(\mathbf{w}) = \sum_{k=1}^{n} t_{kf} \| \mathbf{x}_k - \mathbf{w}_{f(\mathbf{x}_k)} \|^2. \tag{8}$$

The cluster centers $\mathbf{w}$ are obtained by solving the equation

$$\frac{\partial J_{rkm}(\mathbf{w})}{\partial \mathbf{w}} = 0. \tag{9}$$

The existence of the solution to Eq.(9) can be proved easily if the Euclidean distance is assumed. To solve this equation, we first compute an approximate $\mathbf{w}$ with the following equation:

$$\mathbf{w}_j = \frac{\sum_{k=1}^{n} \delta_j(\mathbf{x}_k) t_{kf} \mathbf{x}_k}{\sum_{k=1}^{n} \delta_j(\mathbf{x}_k) t_{kf}}, \tag{10}$$

where

$$\delta_j(\mathbf{x}_k) = \begin{cases} 1 & \text{if } j = f(\mathbf{x}_k) \\ 0 & \text{otherwise} \end{cases}. \tag{11}$$

Then $\mathbf{w}$ can be obtained by applying Newton's algorithm using the result of Eq.(10) as the initial values. Since there is a dependency between the cluster centers and the probability of belongingness, the step one and two are performed iteratively until $\mathbf{w}$ converges.

2.2.4 Redundant cluster deletion

When the assumed number of clusters is greater than the one that a data set really has, there will be some redundant clusters. Such redundant clusters will scramble for the data vectors that should belong to one cluster, thus those data vectors will be divided into two or more clusters forcibly. Since this partition does not match with the natural structure of those data vectors, the clustering will become unstable and sensitive to noise.

To solve this problem, we merge two redundant clusters into one cluster according to their variance and reliability.

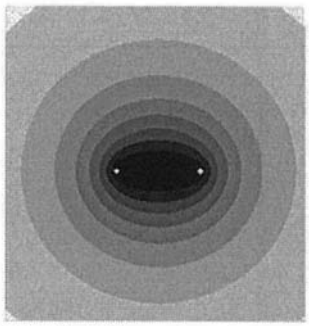


Fig. 7 Reliability field around two cluster centers. The reliability values are showed by intensity.

Fig.7 illustrates a reliability field in a two-dimension space around two cluster centers. The data vectors located on the line connecting the two cluster centers will have higher reliability values than the other data vectors. Such data vectors will have the effect to attract the two cluster centers together, and this effect will become stronger when the two cluster centers really get closer.

Fig.9 shows an experimental result of clustering one crowd of data vectors when given two initial clusters. Here the two cluster centers (red circle) get closer and closer as the iteration increases. Then, the average reliability of the data vectors of the two

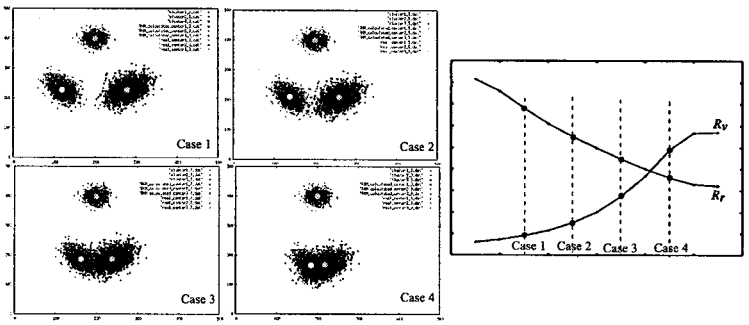
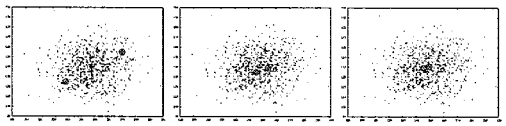


Fig. 8 Investigation on the possibility of merging two clusters.

clusters will become lower and lower according to Eq.(3).



initial state iteration: 5 iteration: 9

Fig. 9 The changes of two cluster centers in one crowd of data vectors during updating.

We considered that it is possible to judge if two clusters should be merged by checking their average reliability and variance. Although the variance of a data set of n dimensional vectors are generally described by its covariance matrix, here we compute a single value to describe the dispersion of the data vectors of cluster i as following:

$$v(i) = \frac{\sum_{\mathbf{x}_k \in \text{cluster}(i)} (\mathbf{x}_k - \overline{\mathbf{w}_i})^2}{N}, \qquad (12)$$

where N is the number of data vectors in cluster i. The average reliability of cluster i can be computed by:

$$r(i) = \frac{\sum_{\mathbf{x}_k \in \text{cluster}(i)} R(\mathbf{x}_k)}{N}. \qquad (13)$$

In order to establish a standard for checking if two clusters should be merged into one big cluster or not, we did some experiments to segment various simulated data set with the method described in subsection 2.2, and analyzed the relation between the average reliability, the dispersion and the distributions of the data vectors.

In **Fig.**8, Case 1~4 show the clustering results of data sets under different distribution. In order to check if two clusters should be merged or not, we compare the average reliability and the dispersion before and after merge. The result of this comparison is obtained by the following equations:

$$\begin{cases} R_v(i,j) & = \frac{v(i)+v(j)}{v(i \cup j)} \\ R_r(i,j) & = \frac{r(i)+r(j)}{r(i \cup j)} \end{cases}, \qquad (14)$$

where $i \cup j$ indicates the merged cluster from cluster i and j.

The graph in the right part of **Fig.**8 shows the results of R_v and R_r of the two clusters. By analyzing these results, we discovered that the ratio of R_r to R_v can be used to judge if two clusters should be merged:

$$Merge(i,j) = P_m(i,j) = \frac{R_r(i,j)}{R_v(i,j)}. \qquad (15)$$

If $P_m(i,j) \leq 1$ then the cluster i and the cluster j should be merged. Then, the redundant cluster deletion can be carried out as the following steps:

1. For each cluster (i), find out its nearest cluster j then check if $P_m(i,j) \leq 1$.
2. If such cluster pairs do not exist, terminate this procedure.
3. Find out the pair that the distance between the two cluster centers is shortest and let i and j denote the number of the smaller and bigger clusters, respectively.
4. Merge the data vectors in the cluster j into cluster i. This can be done by assigning the cluster number i to all the data vectors of cluster j, and using $(\mathbf{w}_i + \mathbf{w}_j)/2$ as the initial cluster center of the new cluster i, which is the cluster merged from cluster i and j.
5. remove the cluster j.
6. repeat step 1.

This procedure should be executed after each update of cluster centers. Therefore, we add the redundant cluster deletion procedure as the third step into the process of data grouping described in subsection 2.2.

2.2.5 The RK-means algorithm

The RK-means clustering algorithm is summarized as:

1) Initialization

i) give the number of clusters c and

ii) give an initial value to each cluster center $\mathbf{w}_i, i = 1, \ldots, c$.

The initialization can be performed manually or by using an external method.

2) Iteration of data grouping

while $\mathbf{w}_i, i = 1, \ldots, c$ do not reach fixed points, Do

i) calculate $f(\mathbf{x}_k)$ and $s(\mathbf{x}_k)$ for each $\mathbf{x}_k$.

ii) update $\mathbf{w}_i, i = 1, \ldots, c$ by solving Eq.(9).

iii) delete redundant cluster with the method described in subsection **2.4**

2.3 Object Tracking with Reliability-based K-means Clustering

By applying the *RK-means clustering* algorithm into object tracking, we bring out the *RK-means tracker*, which is a pixel-wise algorithm that remove the target pixels from the surrounding background one within a search area in each frame of a video sequences.

Correspondently, the target centers are described as $\mathbf{f}_T(i) \quad i = 1 \sim K$, the representative background clusters are represented as $\mathbf{f}_B(j) \quad j = 1 \sim m$, and an unknown pixel is described as $\mathbf{f}_u$.

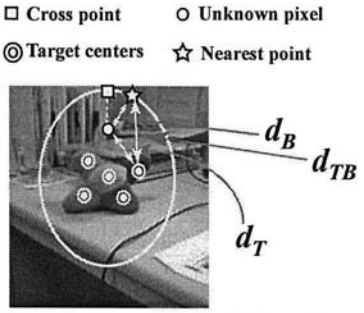


Fig. 10 RK-means clustering on multi-color object with target and background samples.

As shown in **Fig.**10, the classification for an unknown pixel is performed between its nearest target and background clusters. The minimum dissimilarity from $\mathbf{f}_u$ to the target clusters will be calculated as:

$$d_T = \min_{i=1\sim K} \| \mathbf{f}_u - \mathbf{f}_T(i) \|, \tag{16}$$

correspondently, we can get the nearest target cluster center $\mathbf{f}_T(i)$ to $\mathbf{f}_u$. The minimum dissimilarity between $\mathbf{f}_u$ and the surrounding background samples (which are selected from the ellipse contour) is computed as:

$$d_B = \min_{j=1\sim m} \| \mathbf{f}_u - \mathbf{f}_B(j) \|, \tag{17}$$

the dissimilarity between the nearest target center and the nearest background sample ($\mathbf{f}_B(j)$) is calculated as follows:

$$d_{TB} = \| \mathbf{f}_T(i) - \mathbf{f}_B(j) \| . \tag{18}$$

The reliability of the clustering $\mathbf{f}_u$ into target clusters or background clusters is estimated as:

$$R(\mathbf{f}_u) = \frac{d_{TB}}{d_T + d_B}, \tag{19}$$

Then the probability that $\mathbf{f}_u$ belongs to i^{th} the target cluster is calculated as:

$$\mu_T^{(i)}(\mathbf{f}_u) = R(\mathbf{f}_u) * \frac{d_B}{d_T + d_B}. \tag{20}$$

By applying $\mu_T^{(i)}(\mathbf{f}_u)$ into Eq.(10), the new target center $\mathbf{f}_T^{(new)}(i)$ can be calculated. However, if the target centers are only updated in such simple way, one problem will arise: the abrupt color shifting (e.g. high light reflecting) may happen because of the glossy surface of the target object (such as the iris). When such color shifting happens, the feature of the target centers may be completely destroyed. In order to solve this problem, a simple average filter is used to gradually decrease the influence of the rapid color shifting as:

$$\mathbf{f}_T^{(t)}(i) = \gamma \mathbf{f}_T^{(new)}(i) + (1 - \gamma)\mathbf{f}_T^{(t-1)}(i). \tag{21}$$

Where, $0 < \gamma < 1$ is a coefficient, the superscripts $(t), (new), (t-1)$ denote the time. $\mathbf{f}_T^{(new)}(i)$ is the output of Eq(10), $\mathbf{f}_T^{(t-1)}(i)$ is the output of the previous frame, and $\mathbf{f}_T^{(t)}(i)$ is the final result of the present frame. With this equation, when the abrupt color shifting happens, this average filter can resist such influence and when the target color turns back to the original color again, it will continue to update the correct color.

2.4 Update Of The Search Area

After updating the target center, the selected representative background samples are also required to be updated in order to follow the changes of the surrounding background caused by the movement of target object. In this work, since the background samples are selected from the ellipse contour (or it also can be said to be the boundary of search area), the update of background samples is equivalent to that of the search area.

In order to follow the movement and shape changes of the target, the corresponding parameters of the search area are determined according to the distribution of the classified target pixels which are obtained in the current frame. Because the pixels of the target object can be considered as a set of random distributed points, we can represent them by using a probability distribution. Here, the Gaussian probability density function (“***pdf***”) is used to represent the target distribution. The motivation to use the Gaussian *pdf* is:

(1) it is robust to the deformation of the target object;
(2) it can reduce the influence of the mis-detected target pixels caused by the image noise.

Let $\mathbf{S_Z} = [\mathbf{Z}_1, \mathbf{Z}_2, ...\mathbf{Z}_n]^T$ represent the pixel set of the detected target object, where $\mathbf{Z}_i = [x_i, y_i]^T$ is the position of one detected target pixel and n is the number of the pixels belonging to the target object. The Gaussian pdf that describes the distribution of $\mathbf{S_Z}$ can be represented as:

$$\mathbf{S_Z} \sim N(\mathbf{m_Z}, \Sigma_\mathbf{Z}), \tag{22}$$

where, $\mathbf{m_Z}$ is the mean of $\mathbf{S_Z}$ and $\Sigma_\mathbf{Z}$ is the covariance matrix. And they can be described as the following equations:

$$\mathbf{m_Z} = \begin{bmatrix} m_x \\ m_y \end{bmatrix}, \tag{23}$$

$$\Sigma_\mathbf{Z} = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \tag{24}$$

where, m_x and m_y are the centers of the whole detected target pixels in the x and y direction, σ_x and σ_y are the corresponding variances. ρ $(0 < \rho < 1)$ denotes the correlation coefficient of variables. The inverse matrix can be expressed as:

$$\Sigma_\mathbf{Z}^{-1} = \frac{1}{|\Sigma_\mathbf{Z}|}\begin{bmatrix} c_{22} & -c_{21} \\ -c_{12} & c_{11} \end{bmatrix} = \begin{bmatrix} c'_{22} & c'_{21} \\ c'_{12} & c'_{11} \end{bmatrix} \tag{25}$$

With such distribution, in the 2D image coordinate, the Mahalanobis distance of a vector $\mathbf{Z}$ can be expressed as:

$$\begin{aligned} g(\mathbf{Z}) &= [\mathbf{Z} - m_\mathbf{Z}]^T \Sigma_\mathbf{Z}^{-1} [\mathbf{Z} - m_\mathbf{Z}] \\ &= c'_{22}(x - m_x)^2 - 2c'_{12}(x - m_x)(y - m_y) \\ &\quad + c'_{11}(y - m_y)^2 \\ &= c'_{22}x^2 - 2c'_{12}xy + c'_{11}y^2 + 2x(c'_{12}m_y - c'_{22}m_x) \\ &\quad + 2y(c'_{12}m_x - c'_{11}m_y) + c'_{22}m_x^2 + c'_{11}m_y^2 \\ &\quad - 2c'_{12}m_x m_y. \end{aligned} \tag{26}$$

With Eq(26), we can determine an ellipse $E(M)$ that contains M% pixels of the whole target object as:

$$\begin{aligned} g(\mathbf{Z}) &= c'_{22}x^2 - 2c'_{12}xy + c'_{11}y^2 + 2x(c'_{12}m_y - c'_{22}m_x) \\ &\quad + 2y(c'_{12}m_x - c'_{11}m_y) + c'_{22}m_x^2 + c'_{11}m_y^2 \\ &\quad - 2c'_{12}m_x m_y = J, \end{aligned} \tag{27}$$

where, $J = -2ln(1 - \frac{M}{100})$ (as shown in **Fig.11**). When we let M be large enough (e.g. 95), the ellipse $E(M)$ will contain the overwhelming majority of the target pixels.

Comparing with the normal ellipse function

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0, \tag{28}$$

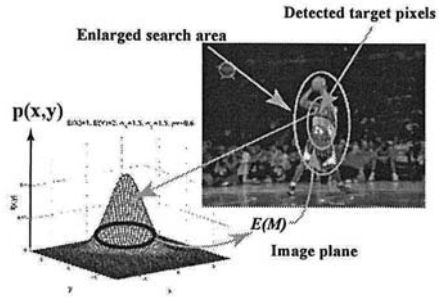


Fig. 11 Ellipse search area determined by the Mahalanobis distance.

we can determine the parameters of the ellipse as:

$$\begin{aligned} &A = c'_{22} \quad B = -c'_{12} \quad C = c'_{11} \\ &D = c'_{12}m_y - c'_{22}m_x \quad E = c'_{12}m_x - c'_{11}m_y \\ &F = c'_{22}m_x^2 + c'_{11}m_y^2 - 2c'_{12}m_x m_y - J. \end{aligned} \tag{29}$$

3. Tracking Failure Detection

In most of the present feature-based tracking algorithms, the target tracking is performed by searching over a small region around the target detected in the previous frame. The advantages of this searching method are:

- reducing the computational cost and making the video-rate object tracking become possible
- improving the robustness of the system when objects similar to the target exit in the image, because they only search over the local region but not whole image.

And the success of such feature-based tracking algorithms is dependent on:.

(1) The tracking result obtained in the previous frame is correct.
(2) The object has not moved out of the search area.

Therefore, after the object tracking has been carried out in the previous frame, a feature-based tracking algorithm is believed to be able to work more robust if it can perform a verification procedure that checks two mentioned elements. In this paper if the tracking failure occurs and is detected by the verification process, a failure recovery process should be carried out. The object tracking will be restarted until the tracking recovery is successfully performed.

As shown in **Fig.12**, the following situation is defined as the tracking failure: in the current frame, the target object moves partially out of the search area that is determined in the previous image, and the original target center contains different color that is similar to the background color. In such case, the tracking failure is regarded as having happened.

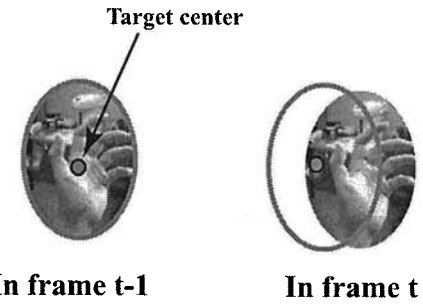


Fig. 12 Definition for the tracking failure.

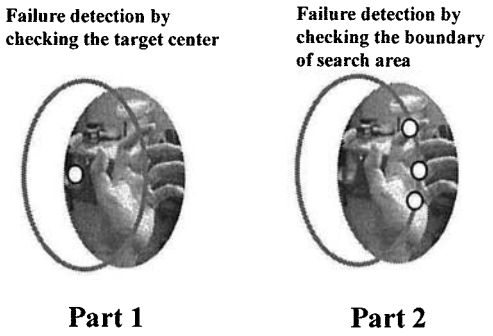


Fig. 13 A perfect tracking failure detection should consist of the failure detection for target centers as well as the background samples.

A correct tracking failure detection is regarded to consist of two parts (as shown in **Fig.**13): 1) tracking failure for the target center; 2) the tracking failure detection for the surrounding background samples. That is because, according to the definition of tracking failure, when tracking failure occurs, some of the background samples on the ellipse contour will absolutely contain the target color (like the Part 2 in **Fig.**13). However, since it is quite possible for the background to contain target color in a random sequence, neither only checking the target centers nor only checking the surrounding background samples will give the reliable detection result. Therefore, we consider a reliable tracking failure detection should be based on checking the target centers as well as the surrounding background samples simultaneously.

3.1 Tracking failure detection for target centers

In this chapter, because the target object is assumed to contain multiple colors, it is nature that the target object consist of multiple target cluster and the tracking failure detection for those cluster center will be performed one by one independently. In the following part of this section, we just explain the examination on one target cluster center.

In frame $t-1$, the i^{th} target cluster is located at $p(i)$ in the 2D space and $\mathbf{c}_{p(i)}^{(t-1)}$ is the target color of this point, $\mathbf{c}_B^{(t-1)}(j)$ is the color contained by the j^{th} background sample. In next frame, the color contained by $p(i)$ will be expressed as $\mathbf{c}_{p(i)}^{(t)}$ and we can perform the tracking failure detection for one target cluster center as:

$$f_T(i) = \begin{cases} 1 & if \quad d_B(i) < d_T(i) \\ 0 & otherwise \end{cases}, \qquad (30)$$

where $d_T(i) = \| \mathbf{c}_{p(i)}^{(t-1)} - \mathbf{c}_{p(i)}^{(t)} \|^2$, $d_B(i) = \min_{j=1\sim m} \| \mathbf{c}_{p(i)}^{(t)} - \mathbf{c}_B^{(t-1)}(j) \|^2$. When $f_T(i) = 1$, it means that in the current frame, the color contained by $p(i)$ looks more like to be the background color rather than the previous target color. And at this time, we will consider that the i^{th} target color may be lost, otherwise $p(i)$ still contains the i^{th} target color.

3.2 Tracking failure detection for background samples

As mentioned in Section10, the ellipse contains the whole target object in the previous frame and should still include the target object in the current frame. Therefore, if tracking failure occurs, the target object will get over the ellipse boundary in the current frame. This may happen when the velocity of the target is greater than the assumed maximum velocity. As shown in the Part 2 of **Fig.**13, this situation can be verified by checking whether the color of the background samples selected from the ellipse contour contains similar color to that of the target color or not.

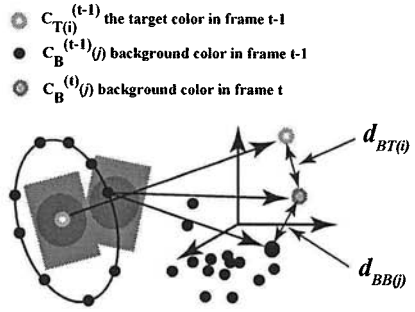


Fig. 14 Tracking failure detection for the surrounding background samples.

As shown in **Fig.**14, $\mathbf{c}_{T(i)}^{(t-1)}$ is the color of the i^{th} target cluster center in the previous frame, $\mathbf{c}_B^{(t-1)}(j)$ is the color of the j^{th} selected background sample in the previous frame. While, $\mathbf{c}_B^{(t)}(j)$ is the color

contained by j^{th} background sample in the current frame. And the tracking failure detection for the selected background samples will be performed:

$$f_{B(j)}^{(i)} = \begin{cases} 1 & if \quad d_{BT(i)} < d_{BB(j)} \\ 0 & otherwise \end{cases}, \qquad (31)$$

here, $d_{BT(i)} = \| \mathbf{c}_{T(i)}^{(t-1)} - \mathbf{c}_B^{(t)}(j) \|^2$ is the dissimilarity between the color of $\mathbf{c}_{T(i)}^{(t-1)}$ and that of $\mathbf{c}_B^{(t)}(j)$; $d_{BB(j)} = \| \mathbf{c}_B^{(t-1)}(j) - \mathbf{c}_B^{(t)}(j) \|^2$ is the dissimilarity between the colors contained by the j^{th} background sample in two adjacent frames. When $f_{B(j)}^{(i)} = 1$, it means that in the current frame, the color of the j^{th} background sample is much more similar to the color of the i^{th} target cluster center (which is determined in the previous frame) than the color contained by the j^{th} background sample in the previous frame, otherwise the j^{th} background sample will be still considered as a normal background point. In such case, we will consider that some parts of the ellipse contour is overlapped by the target object, which indicates that the target object moves out of the search area.

However, if the target object does not move out of the search area and the selected background happens to contain the color that is similar to the target color, only checking one background sample will be unreliable. Therefore, to make the tracking failure detection for background sample reliable, in this step we let $j = 1 \sim 36$. When $\sum_{j=1\sim 36} f_{B(j)}^{(i)} = Threshold$ ($Threshold = 14$ means 40% of the selected point are wrong), some of these selected background samples contain the i^{th} target color in the current, we will say that the failure detection for background samples stands and the target object is quite possible moves out of the search area.

The probability of tracking failure over the multi-color target object can be estimated with the following equation:

$$Pro(failure) = 0.5 * \frac{\sum_{i=1}^{K} f_T(i)}{K} + 0.5 * \frac{\sum_{i=1}^{K}\sum_{j=1}^{36} f_{B(j)}^{(i)}}{K * 36}. \qquad (32)$$

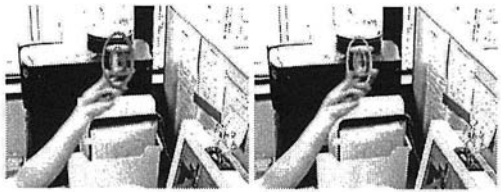
When $Pro(failure)$ is greater than 0.7 (this is the least value when tracking failure happens), we will claim that the tracking failure has occurred. Since all these calculations are very simple, the tracking failure detection process will cost little time and has little affection on the processing speed.

4. Tracking Failure Recovery

After knowing the tracking failure has occurred, the recovery from such tracking failure is naturally required. In order to get the correct new target center, the tracking failure recovery process is performed in the 5D feature spaces. That is because only checking the color information will not be reliable, some components of the background may contain similar color to the target color, and at this time the geometric feature becomes important for recovery.

The previous frame. Tracking failure happens

Fig. 15 Successful tracking failure detection when the colors of both the target and background are changed by the abrupt camera flash. To indicate the failure detection result clearly, we change the ellipse contour to green.



The previous frame. Tracking failure happens

Fig. 16 Successful tracking failure detection. Since the velocity of object is too high, the object moves out of the search area.

In this step, as shown in **Fig.**17, the last target center which is detected in the previous frame is represented as $\mathbf{f}_T^{(t-1)}$. $\mathbf{f}^{(t)}$ means the arbitrary pixel within the original search area (or ellipse), so $\mathbf{f}^{(t)} \in ellipse$. The new recovered target center $\mathbf{f}_T^{(t)}$ is detected by searching over the whole ellipse pixel-by-pixel and finding out the nearest pixel to $\mathbf{f}_T^{(t-1)}$ in the 5D feature space. And this calculation can be expressed as:

$$\mathbf{f}_T^{(t)} = \underset{\mathbf{f}^{(t)} \in ellipse}{argmin} \| \mathbf{f}_T^{(t-1)} - \mathbf{f}^{(t)} \|^2 . \qquad (33)$$

After getting the newly recovered $\mathbf{f}_T^{(t)}$, $\mathbf{f}_T^{(t)}$ will be taken as the new target cluster center, and the original search area will be translated so as to be centered at $\mathbf{f}_T(t)$. Then, the tracking process will be restarted.

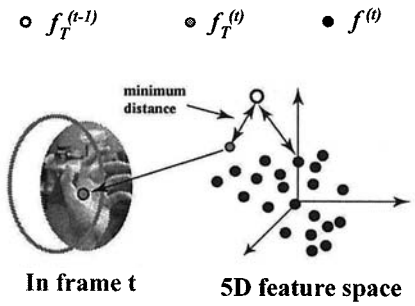


Fig. 17 Failure recovery in the 5D feature space.

5. Verification of the Recovered Target Color

Since the color of a target cluster is updated continuously during tracking, it will be quite possible that the updated target color will be different from the initial one. Therefore, it is very necessary to check whether the color of the recovered target cluster center is correct or not by examining the similarity between the recovered target color and the initial target color. Here, this verification is achieved with a Bayesian formulate to confirm the reliability of the recovered target color as:

$$Pro(\mathbf{c}_{T(k)}|\mathbf{c}_{rec(i)}) = \frac{p(\mathbf{c}_{rec(i)}|\mathbf{c}_{T(k)})P(\mathbf{c}_{T(k)})}{\sum_{j=1}^{K}\{p(\mathbf{c}_{rec(i)}|\mathbf{c}_{T(j)})P(\mathbf{c}_{T(j)})\}}, \tag{34}$$

in Eq(34), $\mathbf{c}_{rec(i)}$ is the recovered i^{th} target color, and $\mathbf{c}_{T(k)}$ is the initial color of an arbitrary k^{th} target cluster. $p(\mathbf{c}_{rec(i)}|\mathbf{c}_{T(k)})$ is the likelihood between $\mathbf{c}_{rec(i)}$ and $\mathbf{c}_{T(k)}$, which is described by a Gaussian normal distribution model. And the mean value of this model is $\mathbf{c}_{T(k)}$ and its color variance is calculated in the previous frame. $P(\mathbf{c}_{T(k)})$ is the prior probability of the k^{th} target color, and here this prior probability is described by the ratio between the area of the k^{th} target color and the whole target area in the previous frame. $Pro(\mathbf{c}_{T(k)}|\mathbf{c}_{rec(i)})$ is the posterior that describes similarity between the i^{th} recovered target color and the initial color of arbitrary k^{th} target color.

By iterating Eq(34) K times, we can get one maximum posterior value and here we assuming the corresponding initial target color is the n^{th} one ($n = 1 \sim K$). When $n = i$, it indicates that the recovered target color is just the lost target color and the failure recovery step is correct. Otherwise, the failure recovery is wrong, the tracking process will be stopped.

6. Radial Sampling

In order to reduce the computational cost of RK-means tracker, one of the most effective method is to reduce the volume of the dataset which will be processed while clustering. Therefore, the sampling technique is required for this purpose.

Random sampling sometimes can be a useful sampling method for object tracking, but since it randomly selects the sample points, the property of the selected sample points is usually un-predictable. The selected sample point may be a target point and also may be a background one. Therefore, the clustering result among the sample points produced by random sampling will become unstable.

In order to keep the correctness of the clustering result, a radial sampling method is used in this research as shown in **Fig.**18.

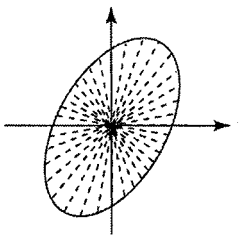


Fig. 18 Illustration of the radial sampling.

The radial sampling is set up according to such an assumption: when the target object is solid and included within the search area, the closer a pixel is to the center of the search area, the more possible it is a target pixel. In this research, because the search area is described by an ellipse model, this assumption becomes: the closer to the ellipse center, the more possible a pixel is a target pixel. In this research, because the search area is updated according to the distribution of the detected target pixels, the target center is the center of search area. Therefore, this assumption is reasonable.

This radial sampling method is performed in the following way: all the sample points are selected from the ellipse radius at constant intervals; each ellipse radius is produced at the intervals of constant degree. This radial sampling is also equivalent to produce a density (or probability) map (which is quite similar to the kernel-weighted map of mean-shift algorithm) where the ellipse center has the highest density, while the pixels near the ellipse contour have low probability to be the target ones (as shown in **Fig.**19). Since there are few sample points selected from the regions far away from the

ellipse center, the affection of noise pixels (or the mixed background pixels) has been reduced, thus improving not only the processing speed but also the robustness of the RK-means clustering.

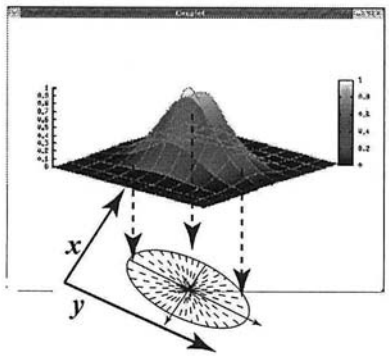


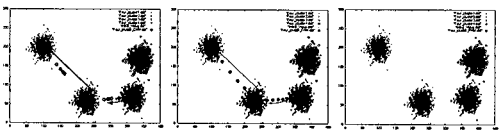
Fig. 19 Radial sampling and the center-weighted density map.

7. Experiment and discussion

For concision, hereafter the Hard (or existing) K-means clustering, Fuzzy K-means and our RK-means are abbreviated as *EKM*, *FKM* and *RKM*, respectively. In all the experiments, we give the same the initial number of clusters and the cluster centers points for the same image sequence, and all the algorithms run at the same iterations for one image frame.

7.1 Evaluating the effectiveness of RK-means

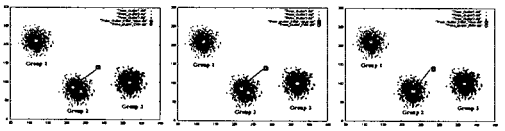
We compared the clustering results of RKM with those of EKM and FKM within the same simulated data sets.



(A) EKM (B) FKM (C) RKM

Fig. 20 Clustering results when an undefined cluster exists.

Fig.20 shows the clustering results when an undefined cluster exists. In this experiment, we gave three initial clusters. In the figure, the sky blue "□" denotes the initial cluster centers and the yellow "•" shows the updated cluster centers during iteration and the resulted cluster centers are the ones pointed by arrows. The EKM only succeeded in setting one cluster center correctly. The FKM also failed to gave good clustering result. Our RKM found three of the four groups of the data vectors successfully and gave the correct cluster centers for them. The data vectors of the fourth group were given extremely low reliability by RKM. This means they can not be classified reliably and should not be assigned to any of given clusters.



(A) EKM (B) FKM (C) RKM

Fig. 21 Clustering results when a redundant cluster exists.

As shown in **Fig.**21, we give four initial clusters for the data set only having three groups of crowded data vectors. The hollow "□" indicates the initial cluster centers, and the solid sky blue "□" shows the resulted cluster centers. The EKM and FKM algorithm brutally divided the data vectors of *group 2* into two separated clusters (**Fig.**21(A) and (B)). Meanwhile, the RKM removed one redundant cluster during the iteration of data grouping and gave only one cluster for that data group (see **Fig.**21 (C)).

7.2 Convergence of the RK-means

One important thing that needs to be confirmed is whether the RKM clustering algorithm converges. We used the IRIS dataset☆ that is a common database often used for testing data grouping algorithms to check the convergence of RKM. The IRIS dataset has 150 data points. It is divided into three groups and two of them are overlapping. Each group contains 50 data points. Each point has four attributes.

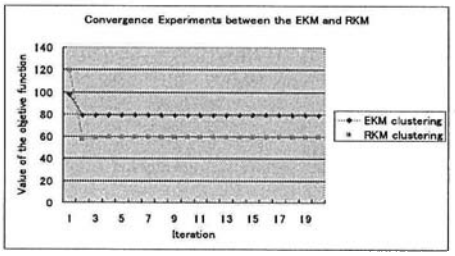


Fig. 22 Convergence of the EKM and the RKM algorithms.

From the experimental result shown in **Fig.**22, we confirmed that the convergence of the RKM algorithm is as good as the EKM algorithm, and the speed of convergence is not slower than the EKM.

7.3 Affection of the cluster volume on RKM

Another important element that will affect the

☆ http://www.ics.uci.edu/~ mlearn/databases/iris/iris.data

performance of RKM is the volume (sometimes the size) of each cluster.

Fig.23 shows a case that both the distribution and volume of one cluster increase. Here, the distribution of the left cluster is a circle, and both its distribution and volume are fixed, the distribution of the right cluster is an ellipse and its distribution will increase as its volume increase. In the graph of **Fig**.23, the horizontal axis shows the ratio between the volume of each cluster (and this ratio will be increased larger and larger), the vertical axis is the average reliability of each cluster. The red line shows the changes of the average reliability of the left cluster and the green one shows that of the right cluster. From this image, it is obvious that the increase of the distribution and volume of one cluster has very little affection on the average reliability of each cluster calculated by the RKM. In other words, the RKM is insensitive to the changes of volume and distribution of clusters.

Fig.24 shows the comparative experiment on the displacement between the EKM and RKM in the case of **Fig**.23. The red line shows the displacement of the EKM and the green line shows that of the RKM. The left graphics shows the comparative results of RKM and EKM on the left cluster of **Fig**.23, correspondently, the right graphics shows the results on the right cluster. According to both graphics, obviously the RKM works better than the EKM in this case. That is because the EKM will mistakenly take some outliers as the pixels belonging to some clusters. While the RKM can give low reliability to such outliers.

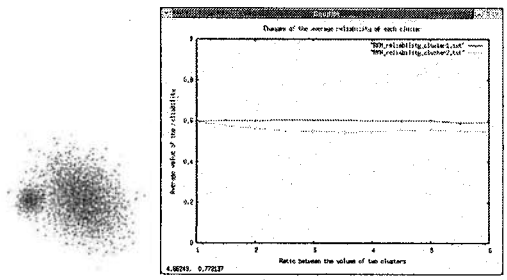


Fig. 23 Experiment of the RK-means clustering under variable distribution. In the right graph, the horizontal axis shows the ratio between the volumes of two clusters, the vertical axis shows the average reliability of each cluster. The red curve shows the average reliability of the left cluster, and the green one shows that of the right cluster.

Fig.25 shows the result of an image segmentation experiment for testing the effect of the redundant cluster deletion. **Fig**.25(A) was extracted from

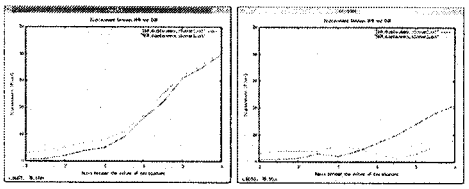


Fig. 24 The displacements between the EKM and RKM on each cluster in the case of **Fig**.23. Left graph: on the left cluster; Right graph: on the right cluster. Green curve: the displacement of RKM; Red Curve: the displacement of EKM

(A) (B) (C) (D)

Fig. 25 Image segmentation with redundant cluster deletion. A: Input Image; B: Iteration 1; C: Iteration 2; D: Iteration 3.

frame 285 in an image sequence used for the experiments shown in **Fig**.28. The yellow cross indicates the target cluster centers. The target clusters become less during the iteration of data grouping, as shown in (B)~(D). These results show that the RKM algorithm can automatically delete the redundant clusters.

7.4 Object tracking with RK-means tracker

7.4.1 Initialization

In order to achieve the robust tracking result, the initial target centers are manually selected in this research.

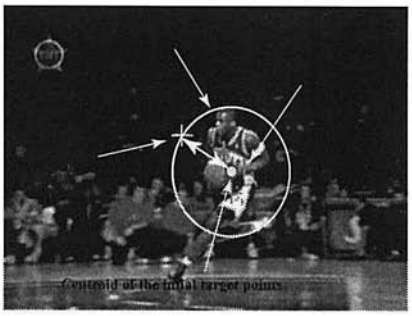


Fig. 26 Initialization for multi-color object tracking.

As shown in **Fig**.26, in the first image, K points are manually selected as the initial target points, where the number of K depends on the main colors contained by the target object. Then, the centroid of these initial target points will be calculated and be regarded as the center of the initial search area. Thirdly, one initial background point out of the target object will be selected. Finally, an initial

search area will be constructed, which is a circle in the first image. The center of search area is located at the centroid of the initial target points and its radius is the distance between the initial background point and the center of circle.

7.5 Object tracking with RK-means tracker

Hua *et al.* [13] have compared[☆] K-means tracker with some famous tracking algorithms. Here we only compare our RK-means tracker with the K-means tracker.

Fig.27 shows the experimental result of RK-means tracker in the case that the target person disguises himself while tracking. In this experiment, the target person uses the sunglass, towel and the hat to changes his appearance. Because the RK-means clustering is applied between the target and background samples, the RK-means tracker can distinguish such objects from the target. By updating the search area dynamically, the RK-means tracker can also deal with the rotation of target object.

Fig.28 shows a sequence of hand tracking. As for the K-means tracker, the tracking failed since frame 285. In frame 285, since the color of some surrounding background parts (e.g. a corrugated carton) was similar to the skin color for some degree, the k-means tracker mistakenly took them as the target pixels. This caused the update of the search area to fail, so did the tracking. As for our RK-means tracker, in frame 285, the influence of the background parts having color similar to the skin color was effectively repressed through the reliability evaluation. As the result of it, the RK-means tracker could detect the target area (e.g. hand) and update the search area correctly.

Fig.29 shows the performance of RK-means tracker when the target person is fighting against another person[☆☆]. Since two persons are grabbling with each other, the other person is mixed into the search, which is difficult for the conventional algorithms to work correctly. But the RKM algorithm can successfully discriminate the target person from another one, because the pixels of another person has low reliability to be considered as the target ones.

All the experiments were performed with a desktop PC with a 3.06GHZ Intel XEON CPU, and the image size was 640×480 pixels. When the target size varied from $100 \times 100 \sim 200 \times 200$ pixels, the processing time of our algorithm was about $9 \sim 15$ms/frame.

[☆] http://vrl.sys.wakayama-u.ac.jp/VRL/studyresult/study_result_3_en.html

[☆☆] The video is taken from PETS-ECCV2004.

8. Conclusion

In this paper, we have proposed a robust pixel-wise object tracking algorithm which is based on a new reliability-based K-means clustering algorithm (called as RK-means tracker). By applying the RK-means clustering into both the target and background samples, the RK-means tracker can give low reliability value to both the background and noise pixels to delete them, thus it achieves the robust object tracking under the cluttered condition. By dynamically updating the search area according to the distribution of the target pixels, the RK-means tracker can follow the target deformation smoothly. When the target is partial lost, the tracking failure detection and recovery processes are brought out to solve this problem. Through the extensive experiments, we have confirmed that the effectiveness and advantages of the RK-means tracker.

Acknowledgments This research is partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant in-Aid for Scientific Research (A)(2), 16200014 and (C) 18500131, (C) (2) 15500112.

References

1) J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithm, Plenum Press, New York, 1981
2) R. Krishnapuram and J.M. Keller, A Possibilistic Approach to Clustering, *IEEE Trans. Fuz. Sys.*, Vol. 1, No. 2, pp.98-110, 1993.
3) Chintalapudi K.K., Kam M., A noise-resistant fuzzy C means algorithm for clustering, *FUZZ-IEEE*, Vol.2, pp.1458-1463, 1998.
4) J.M. Jolion, etc., Robust Clustering with Applications in Computer Vision, *PAMI*, vol. 13, No. 8, 1991.
5) Ron Zass and Amnon Shashua A Unifying approach to Hard and Probabilistic Clustering, *ICCV*, Vol 1, pp.294-301, 2005.
6) J.Hartigan, M.Wong, Algorithm AS136: A K-means clustering algorithm, *Applied Statistics*, vol.28, pp.100-108, 1979.
7) N.Peterfreund, Robust tracking of position and velocity with Kalman snakes, *PAMI*, Vol.22, pp.564-569, June 2000.
8) C.Gräβl, etc., Illumination Insensitive Template Matching with Hyperplanes,*DAGM*, pp.273-280, 2003
9) M.Isard and A.Blake, CONDENSATION-Conditional density propagation for visual tracking, *IJCV*, Vol.29, No.1, pp.5-28, 1998.
10) K.Toyama, A.Blake, Probabilistic Tracking in a

Fig. 27 Disguise experiment. The RK-means tracker is applied to track the human head where the target person disguises himself.

Metric Space,*ICCV*, Vol.2, pp.50-57, 2001

11) A.D.Japson, etc., Robust Online Appearance Model for Visual Tracking, *PAMI*, Vol.25, No.10, 2003

12) D.Comaniciu, V.Ramesh and P.Meer: Kernel-Based Object Tracking,*PAMI*, Vol.25, No.5, pp.564-577, 2003

13) C.Hua, H.Wu, T.Wada, Q.Chen, K-means Tracking with Variable Ellipse Model, *IPSJ Transactions on CVIM*, Vol.46, No.Sig 15(CVIM12), pp.59-68, 2005

Results of the K-means tracker. The missing target center and similar color of background make it failed since Frame 285.

Results of the RK-means tracker.

Frame 220 Frame 258 Frame 285 Frame 305

Fig. 28 Results of comparative experiment with K-means tracker [13] and RK-means tracker under complex scenes.

Frame 055 Frame 075 Frame 107 Frame 117

Frame 125 Frame 138 Frame 142 Frame 168

Fig. 29 Applying the RKM into object tracking. This video is taken from the public database of PETS-ECCV2004, and in this sequence the target person is fighting against another person. The RKM-based tracking system has successfully tracked the target person against the other person.

14) R. Collins and Y. Liu: "On-line Selection of Discriminative Tracking Feature", *ICCV*, Vol.2 pp.346-352 2003

15) H.T.Nguyen and A. Semeulders: "Tracking aspects of the foreground against the background", *ECCV*, Vol.2 pp.446-456 2004

16) A. Rosenfeld and A.C. Kak: " Digital Picture Processing, Computer Science and Applied Mathematics", *Academic Press*, New York, 1976

17) Stauffer, C. and Grimson, W.E.L, "Adaptive Background Mixture Models for Real-time Tracking", *CVPR*, pp.246-252, 1999

18) J.Barron, D.Fleet, S.Beauchemin: "Performance of optical flow techniques", *IJCV*, Vol.2, No.1, pp. 42-77, 1994

19) C.Zhang, Y.Rui, "Robust Visual Tracking via Pixels Classification and Intergration", *ICPR*, pp. 37-42, 2006