Proceedings of the 2004 IEEE
International Conference on Networking, Sensing & Control
Taipei, Taiwan, March 21-23, 2004

On the Distribution-Based Tracking Systems

Hwann-Tzong Chen[1,2] Tyng-Luh Liu[1] Chiou-Shann Fuh[2]

[1]Institute of Information Science, Academia Sinica, Nankang, Taipei 115, Taiwan
[2]Department of CSIE, National Taiwan University, Taipei 106, Taiwan

Abstract

We investigate the issues of object representation and search techniques for distribution-based tracking systems. While representing objects by color distributions has the advantage to capture the essential portion of a tracked object, it generally does not handle scale changes appropriately. We thus adopt a new object representation by integrating color and edge information via two coupled weighting schemes derived from a covariance ellipse model. The representation allows a system to perform optimization over a continuous space and to yield better tracking performances. On the aspect of search techniques, we discuss two popular iterative optimization approaches: line-search and trust-region methods. We demonstrate the differences of the two by analyzing the quality of their respective solutions through numerical and real tracking examples.

Keywords: Vision, tracking, trust-region methods.

1. Introduction

Visual tracking is an important topic with practical applications in vision research. There are quite a number of methods proposed over the years. Nevertheless, our review focuses only on distribution-based tracking systems.

1.1. Related work

If the objects to be tracked are non-rigid, it is convenient to represent them with probability distributions of some salient features of the objects. The most straightforward way to derive a distribution model is through *histogram analysis* [1], [2], [3], [4]. Birchfield [1] has proposed an algorithm for tracking a person's head by modeling it as a vertical ellipse with a fixed aspect ratio. In [2], Bradski presents a CAMSHIFT (continuously adaptive mean shift) system for use in a perceptual user interface to track face. Comaniciu et al. [4] apply mean shift analysis to real-time tracking for non-rigid objects. They model objects by color distributions, which are constructed via *kernel density estimation*, and then measure the similarity between the target and candidate distributions using a Bhattacharyya coefficient. The best location that maximizes the similarity measure is found iteratively by moving along the direction of a mean shift vector, which approximates the density gradient of the Bhattacharyya coefficient.

More recently, Perez et al. [7] use the Bhattacharyya coefficient to compare two color distributions. They incorporate this similarity measurement of color distributions into the observation likelihood of a probabilistic framework, and can then apply the particle filter technique to color-based tracking. Also, in [9], Zhang and Freedman derive a PDE-based curve flow that describes the evolution of an object's contour. The curve flow is guided by the likeness between the candidate and target color distributions. The likeness can be measured by either the Bhattacharyya coefficient or by the Kullback-Leibler divergence.

2. Distribution-Based Tracking Systems

Tracking objects by distributions is efficient but not necessarily sufficient if appreciable amount of changes in scale and shape are present. Suppose that the scale of a target object of monotone color is enlarged. Then, it is not always guaranteed that the appropriate scale will be recovered since, in this case, any sub-portion of the object has a similar distribution to that of the object, i.e., the optimal solutions are not unique. Therefore other tracking cues are needed to elevate the performance of a distribution-based tracker, e.g., [8]. In our system, the representation model consists of two elements: the first one is to characterize the RGB color distribution, and the second is to estimate the edge point density near the boundary of an object. Since the latter is contributed only from samples near the boundary, and more prone to be affected by the scene background, especially in clutter, its significance is weighted less to make the color distribution the primary cue for determining a target object's whereabouts.

2.1. Representation models

Color distribution: We divide the RGB color space into n bins to model the color distribution. A single-valued bin assignment function b is defined uniquely by pixel's RGB value as $b : \mathbf{x}_i \mapsto \{1, \dots, n\}$, where $\mathbf{x}_i$ is any pixel in an image. To account for non-rigidity, a weighting scheme based on the *bivariate normal distribution* is adopted so that color features at different locations within an ellipse are treated differently. In particular, we use $\varepsilon(\mathbf{x}; \boldsymbol{\zeta}) = 1$ to represent an ellipse centering at $\boldsymbol{\mu} = (\mu_1, \mu_2)$, where $\boldsymbol{\zeta} = (\boldsymbol{\mu}, \boldsymbol{\sigma}, \rho)$ is a five-dimensional vector, and $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ and ρ are related to the lengths of the principal semi-diameters and the rotation angle with respect to the horizontal axis. Then, the area within the ellipse can be represented as $A_1(\boldsymbol{\zeta}) = \{\mathbf{x} \mid \varepsilon(\mathbf{x}; \boldsymbol{\zeta}) \le 1\}$, where the subscript 1 is to emphasize that each such an ellipse is indeed the *covariance ellipse* of a bivariate normal distribution.

To compute a probability distribution of color for tracking, let I^0 be the first image frame and $\boldsymbol{\zeta}^0 = (\boldsymbol{\mu}^0, \boldsymbol{\sigma}^0, \rho^0)$. Furthermore, a target object initially centering at $\boldsymbol{\mu}^0$ can be associated with $A_1(\boldsymbol{\zeta}^0) = \{\mathbf{x} \mid \varepsilon(\mathbf{x}; \boldsymbol{\zeta}^0) \le 1\}$, the area enclosed by the corresponding covariance ellipse, $\varepsilon_1(\boldsymbol{\zeta}^0)$. We then define the target's color distribution within $A_1(\boldsymbol{\zeta}^0)$, denoted as $p(u; \boldsymbol{\zeta}^0)$, by

$$p(u; \boldsymbol{\zeta}^0) = \frac{1}{C_p} \sum_{\mathbf{x}_i \in A_1(\boldsymbol{\zeta}^0)} w_c(\mathbf{x}_i; \boldsymbol{\zeta}^0) \delta(b(\mathbf{x}_i) - u),$$

where δ is the Kronecker delta function, and w_c is the weighting function for color distribution derived from the bivariate normal, i.e.,

$$w_c(\mathbf{x}_i; \boldsymbol{\zeta}^0) = \exp\left\{ -\frac{\varepsilon(\mathbf{x}_i; \boldsymbol{\zeta}^0)}{2} \right\}. \qquad (1)$$

That $p(u; \boldsymbol{\zeta}^0)$ is a probability implies that the total weight $C_p = \sum_{\mathbf{x}_i \in A_1(\boldsymbol{\zeta}^0)} w_c(\mathbf{x}_i; \boldsymbol{\zeta}^0)$. For convenience, the notation $p(u; \boldsymbol{\zeta}^0)$ will be abbreviated into $p(u)$ since $\boldsymbol{\zeta}^0$ only describes the target's initial state. Analogously, during tracking an image area enclosed by some $A_1(\boldsymbol{\zeta})$, its color probability distribution, denoted as $q(u; \boldsymbol{\zeta})$, will be

$$q(u; \boldsymbol{\zeta}) = \frac{1}{C_q} \sum_{\mathbf{x}_i \in A_1(\boldsymbol{\zeta})} w_c(\mathbf{x}_i; \boldsymbol{\zeta}) \delta(b(\mathbf{x}_i) - u),$$

where C_q is the total weight such that $\sum_{u=1}^{n} q(u; \boldsymbol{\zeta}) = 1$.

Edge density: The motivation for estimating the edge density near the loci of a covariance ellipse is to aid the system to determine a reasonable solution if there are several competing covariance ellipses of similar color distributions. We first use a high-pass filter with a typical 5×5 Laplacian kernel to perform convolution and to generate a binary edge map E for each image frame [6].

Just like the use of a color weighting function w_c for the color distribution, an edge weighting function w_e is called for to derive an appropriate edge density estimation near an ellipse's loci. Conveniently, for every w_c defined by a covariance ellipse in (1), we can define a corresponding w_e for edge weights using a coupled *crater function*, i.e.,

$$w_e(\mathbf{x}_i; \boldsymbol{\zeta}) = \gamma\, \varepsilon(\mathbf{x}_i; \boldsymbol{\zeta}) \exp\left\{ -\frac{\gamma}{2} \varepsilon(\mathbf{x}_i; \boldsymbol{\zeta}) \right\}, \qquad (2)$$

where γ is the parameter to adjust the shape of a crater function and the size of a crater's *opening*. In practice, we find better tracking performance can be achieved by using a slightly larger γ, say $\gamma = 4$, so that significant values of edge weight are within the covariance ellipse. Finally, we adopt the notation $e(\boldsymbol{\zeta})$ to represent the edge density within and near the boundary of a covariance elliptic region $A_1(\boldsymbol{\zeta})$. The *scale-invariant* definition of $e(\boldsymbol{\zeta})$ is as follows.

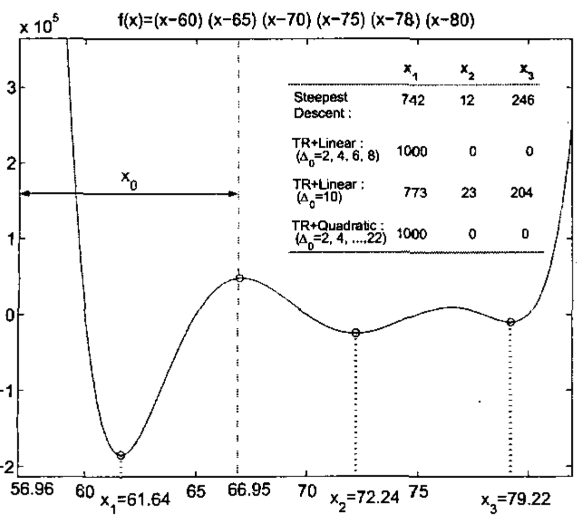
$$e(\boldsymbol{\zeta}) = \frac{1}{\sigma_1 \sigma_2} \sum_{\mathbf{x}_i \in A_1(\boldsymbol{\zeta})} w_e(\mathbf{x}_i; \boldsymbol{\zeta}) E(\mathbf{x}_i), \qquad (3)$$

where $E(\mathbf{x}_i) = 1$ if $\mathbf{x}_i$ is an edge point, and 0, otherwise.

2.2. Iterative optimization techniques

Iterative algorithms for optimization can be divided into two classes: *line-search* and *trust-region*, depending on how they find out the iterates. For a line-search one, the iterates are determined along some specific directions, e.g., the *steepest descent* locates its iterates by considering the gradient directions. A trust-region method, however, derives its iterates in a more general manner by solving the corresponding optimization problem in a bounded region iteratively so that there are more options to select the iterates. In fact, line-search methods can be considered as special cases of trust-region methods. Essentially, there are three elements of any trust-region methods: (i) *trust-region radius*, to determine the size of a trust region, (ii) *trust-region subproblem*, to approximate a minimizer in the region, and (iii) *trust-region fidelity*, to evaluate the accuracy of an approximating solution. (Interested readers may refer to [5] for a detailed discussion on trust-region methods.)

Trust-region vs. line-search: Both trust-region and line-search are guaranteed to converge to a local minimum. However, not all local minima are of interest for real application. It can be shown that typical line-search, e.g., *steepest descent* or trust-region with linear model may often converge to a local minimum that is even *inferior* to a more nearby one. Unlike steepest descent, the mean-shift technique in [4] is a more conservative line-search that instead



	x_1	x_2	x_3
Steepest Descent :	742	12	246
TR+Linear : (Δ_0=2, 4, 6, 8)	1000	0	0
TR+Linear : (Δ_0=10)	773	23	204
TR+Quadratic : (Δ_0=2, 4, ...,22)	1000	0	0

Figure 1. Optimizing with steepest descent, TR+*linear* model, and TR+quadratic model. There are totally three local minima, x_1, x_2, and x_3. Out of 1000 runs, with initial positions x_0s, sampled uniformly from [56.96, 66.95], we record in each entry the number of times that a method converges to a local minimum. Note that even the x_0s are near the best local minimum x_1, a well-known line-search method like steepest descent fails to converge to x_1 258 times.

of taking largest/steepest steps along gradients, it usually progresses by small steps, computed from the information within *fixed-size* windows. Such approach tends to converge to a nearby local minimum whether it is significant or not. Thus both a more sophisticated model approximation and a mechanism to adjust the regions of interest iteratively are needed to reduce the chance of converging to a local minimum not of interest.

In Fig. 1, we construct an objective function with three local minima, x_1, x_2, and x_3. Among them, x_1 is clearly the global minimum. We then test the three schemes: steepest descent, TR with linear model, and TR with quadratic model, using 1000 different initial positions x_0s sampled uniformly from [56.96, 66.95]. Though the x_0s are near the best local minimum x_1, a well-known line-search method like steepest descent could fail to converge to x_1 258 times. On the other hand, trust-region methods are more successful in converging to x_1. For trust-region with linear model, since its performance relies on the ability to adjust trust regions adaptively, the outcomes depend more on the values *of initial trust-region radius* Δ_0. While, with a quadratic model approximation, a trust-region method gains additional information from a better approximation to the objective function, and thus less sensitive to the values of Δ_0.

3. Distribution-Based Objective Functions

Since there are two rather distinct features included in the representation model: *color* and *edge*, the resulting objective function must address the two factors justifiably.

3.1. KL distance and BH coefficient

First, for measuring the similarity between two color distributions, we consider the *Kullback-Leibler* (KL) *distance*,

$$f_c(\zeta) = \sum_{u=1}^{n} p(u) \log \frac{p(u)}{q(u;\zeta)}, \tag{4}$$

where $p(u)$ is the target (true) color distribution and $q(u;\zeta)$ is the one for the covariance ellipse $\varepsilon_1(\zeta)$. Second, to estimate whether the boundary edge density of a candidate covariance ellipse is comparable to the one of target's, we embed the edge density ratio, denoted as $h(\zeta)$, into a *sigmoid function* to derive the following,

$$\begin{aligned} f_e(\zeta) &= 1 - \frac{1}{1+\exp\{-\alpha(h(\zeta)-\beta)\}} \\ &= \frac{1}{1+\exp\{\alpha(h(\zeta)-\beta)\}}, \end{aligned} \tag{5}$$

where $h(\zeta) = e(\zeta)/e(\zeta^0)$; α and β are parameters for setting up the initial sigmoid function. (We have used $\alpha = 5$ and $\beta = 1$ for all the experiments.) Finally, with the definitions in (4) and (5), the underlying optimization problem for image frame I^t can be formally written as

$$\zeta^t = \underset{\zeta\in\Omega^t}{argmin}\, f(\zeta) = f_c + \lambda f_e, \tag{6}$$

where λ is a parameter to weigh the relative importance of the two terms, and Ω^t denotes the space consisting of all possible ζ's for any combinations of translation, scale, and orientation. The implication of (6) is that using color distribution alone, the tracking problem may not always be well-posed, and by coupling with edge density estimation around the boundary, the additional tracking cue functions as an auxiliary term to help yield more appropriate results.

In [4], the Bhattacharyya (BH) coefficient, defined by $f_B(\mathbf{x}) = \sum_{u=1}^{n} \sqrt{p(u)q(u;\mathbf{x})}$, is chosen to be the objective function to be maximized. Since the values of Bhattacharyya coefficient do not vary much (between 0 and 1) and are derived by taking square root, it generally gives a smoother level surface than that of the Kullback-Leibler distance. However, as we will discuss in the next section, a level surface with less variations may cause an iterative optimizer to stuck in a flat region, and fail to perform the optimization properly. In fact, the Kullback-Leibler distance and the Bhattacharyya coefficient are closely related by observing their respective first derivatives,

$$\frac{\partial f_c}{\partial \zeta_i} = -\sum_{u=1}^{n} \frac{p(u)}{q(u;\zeta)} \frac{\partial q(u;\zeta)}{\partial \zeta_i} \tag{7}$$

$$\frac{\partial f_B}{\partial \zeta_i} = \frac{1}{2}\sum_{u=1}^{n} \sqrt{\frac{p(u)}{q(u;\zeta)}} \frac{\partial q(u;\zeta)}{\partial \zeta_i}. \tag{8}$$

3.2. How to compute the KL distance

Each time our tracker is to determine the object's status, it seeks for an optimal elliptic region by minimizing the objective function $f = f_c + \lambda f_e$. While the computation of f_e is straightforward, it requires some efforts to compute the Kullback-Leibler distance f_c for measuring the correlation between two color distributions and the derivatives of f_c for solving the optimization problem.

In practice, many of the color bins would have null distributions in either $p(u)$ or $q(u;\zeta)$. These null values can cause singularities in the denominators of (4). A common way to avoid such singularities is to simply add a small positive number to both $p(u)$ and $q(u;\zeta)$ for all $u \in \{1,\ldots,n\}$, and then re-normalize them into probabilities. However, we find such approach is very unreliable, and often results in misleading results: in particular, it will affect the derivatives of the Kullback-Leibler distance considerably. As in our implementation, we compute a Kullback-Leibler distance by the following approximation:

$$f_c(\zeta) = \sum_{u=1}^{n} p(u) \log \frac{p(u)}{q(u;\zeta)} \approx \sum_{u\in\tilde{U}} \tilde{p}(u) \log \frac{\tilde{p}(u)}{\tilde{q}(u;\zeta)}, \tag{9}$$

where

$$\begin{aligned}
\tilde{U} &= \{u \mid p(u) \neq 0 \text{ or } q(u;\zeta) \neq 0, u = 1\ldots n\},\\
n_p &= \#\text{ of } \{u \mid p(u) = 0, u \in \tilde{U}\},\\
n_q &= \#\text{ of } \{u \mid q(u;\zeta) = 0, u \in \tilde{U}\},\\
p^+_{min} &= \min\{p(u) \mid p(u) > 0\},\\
q^+_{min} &= \min\{q(u;\zeta) \mid q(u;\zeta) > 0\},\\
\tilde{p}(u) &= \begin{cases} (1 - n_p\epsilon_p)p(u), & \text{if } p(u) > 0,\\ \epsilon_p, & \text{if } p(u) = 0,\end{cases}\\
\tilde{q}(u;\zeta) &= \begin{cases} (1 - n_q\epsilon_q)q(u;\zeta), & \text{if } q(u;\zeta) > 0,\\ \epsilon_q, & \text{if } q(u;\zeta) = 0.\end{cases}
\end{aligned}$$

Note that $\epsilon_p = \epsilon \times p^+_{min}$, $\epsilon_q = \epsilon \times q^+_{min}$, and we have $\epsilon = 10^{-5}$ for all experiments. The value of ϵ should be considered together with n, the total number of bins divided for modeling the color distribution, to ensure that

$$\begin{aligned}
n_p \times \epsilon \times p^+_{min} \ll 1 &\implies \tilde{p}(u) \approx p(u),\\
n_q \times \epsilon \times q^+_{min} \ll 1 &\implies \tilde{q}(u;\zeta) \approx q(u;\zeta).
\end{aligned}$$

It follows that $\sum_{u\in\tilde{U}} \tilde{p}(u) = 1$ and $\sum_{u\in\tilde{U}} \tilde{q}(u;\zeta) = 1$, and more importantly, the modified color probability distributions $\tilde{p}(u)$ and $\tilde{q}(u;\zeta)$ differ negligibly from the true distributions $p(u)$ and $q(u;\zeta)$, respectively. The main advantage of the approximation in (9) is that the computation of the Kullback-Leibler distance and its partial derivatives involves only those color bins that either $p(u)$ or $q(u;\zeta)$ has nonzero distribution.

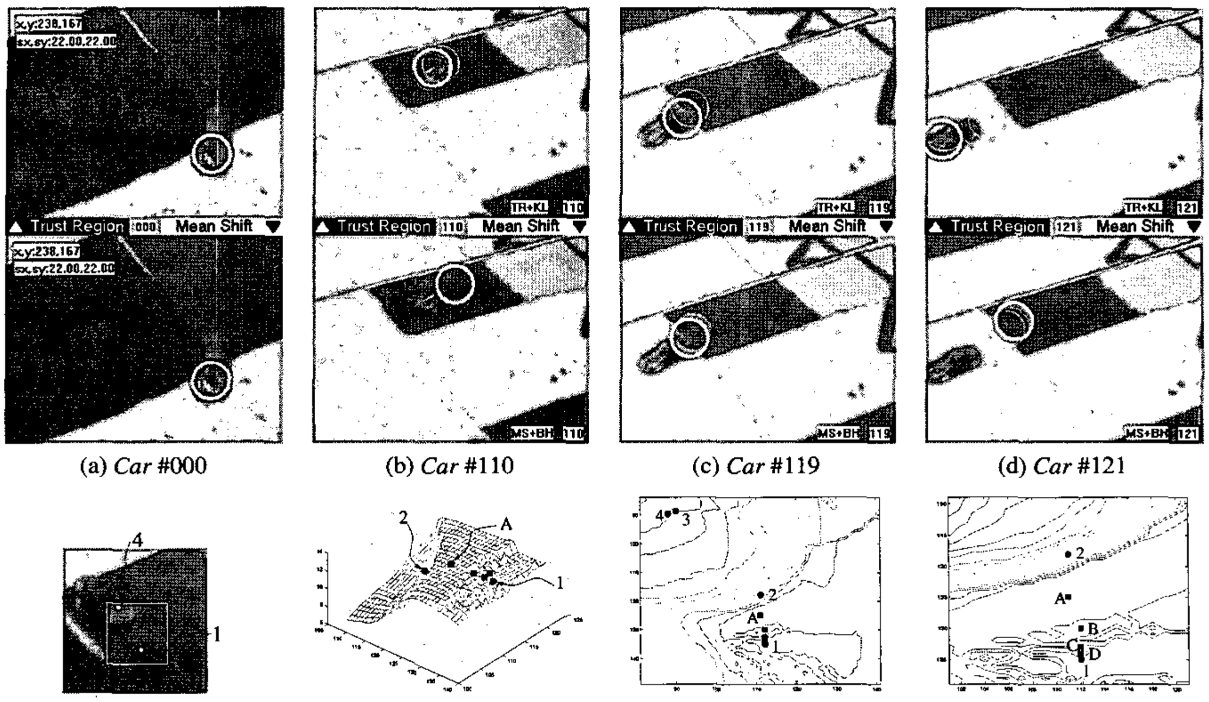
4. Comparisons and Experimental Results

In this section, we demonstrate the efficiency and the reliability of a trust-region tracker by (i) making comparisons with a tracker based on line-search, e.g., the mean-shift tracker by Comaniciu et al. [4], and (ii) investigating the need of a combined objective function (6) and the advantages of using a quadratic model.

4.1. Trust-region vs. mean-shift

In [4], color distribution is used as the only cue for tracking, and the Bhattacharyya coefficient, defined by $\sum_{u=1}^{n} \sqrt{p(u)q(u;\mathbf{x})}$, is chosen to be the objective function to be maximized. Since a mean-shift vector is simply to approximate the gradient of an objective function, thus for the sake of comparison, we implement a trust-region tracker with a linear model approximation, and also use the exact color representation model described in [4] for all comparisons here. This implies we are dealing with two trackers: trust-region (TR) and mean-shift (MS), and two objective functions: Kullback-Leibler distance (KL) and Bhattacharyya coefficient (BH). Totally there are four possible combinations: MS+BH, TR+BH, MS+KL, and TR+KL.

We test the two trackers with the *Car sequence* (see Fig. 2). The catch of this sequence is that the toy car will later run over a portion of the carpet, which is of similar color to the car's, and then complicates the tracking task. The experiments are done first with the Bhattacharyya objective function where the results for both trackers are somewhat less satisfactory mostly due to a flat and smooth level surface, say, when processing the 100th frame of the *Car sequence*. Alternatively, the corresponding KL level surface, shown in Fig. 2f, is more *informative* in the sense that though the level surface is less smooth, it provides more variations and responses owing to the KL formula to correlate two distributions. When the KL is coupled to a trust-region tracker, it again produces consistent and satisfactory tracking results. Using Fig. 2e to explain pictorially,



(a) *Car* #000 (b) *Car* #110 (c) *Car* #119 (d) *Car* #121

(e) A car (#034) (f) KL Level surface (g) KL Level curves (h) Locally enlarged

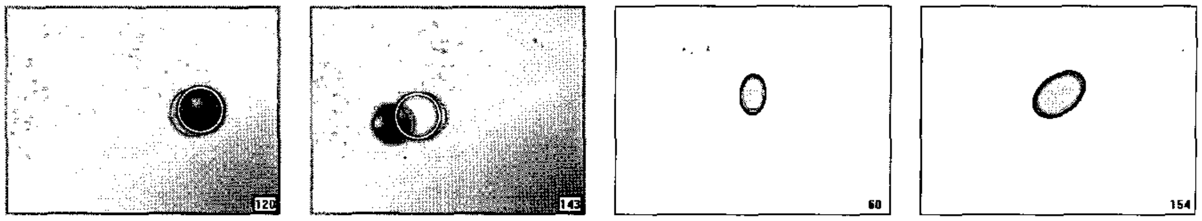
Figure 2. TR vs. MS: (a)-(d) Tracking outcomes of TR+KL and MS+BH are plotted together, top and bottom, respectively. (e)-(f) Comparisons between TR+KL and MS+KL.

the two points, 1 and 4 inside the small rectangle, are the car's positions at previous and current frame, respectively. For a system to track the target correctly, its tracker should move/iterate from point 1 to point 4. In Fig. 2g, the KL level curves within the small rectangular area are plotted, where a magnified portion is provided in Fig 2h. When a trust-region tracker is used, it takes 3 iterations to converge to point 4 ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4$). Instead, a mean-shift tracker will first reach point A (see Fig 2h), then find out the KL value there is higher and iterate backward by $A \rightarrow B \rightarrow \cdots \rightarrow 1$, where in this example all the backward iterates happen to have higher KL level values. As a result, the mean-shift tracker will stay at point 1, and miss the target.

From the first-derivative equations in (7), (8) and the above comparisons, we conclude that the two functions, KL and BH, are closely related, and in most cases, appropriate for the tracking application. However, the KL distance tends to be more responsive that it can detect the differences in the correlated signals and yield a level surface to reflect the variations. Such property is especially useful for an optimization-based tracker, as we have demonstrated in the experiments.

4.2. Tracking by edge density

Apparently tracking by edge density alone is not going to work very well, but it is instructive to understand its characteristics and limitation. In the first experiment, the task is to track an object's contour and to investigate the impact of the speed of object's movement on the tracking outcomes. We have learned when the background is simple, i.e., edges from the background are insignificant, and the object's motion is slow, a trust-region tracker using only the edge density function f_e, defined in (5), can track the contour successfully (see Fig. 3a). The performance then starts to deteriorate once the object speeds up its movement. A typical scenario is shown in Fig. 3b that the tracker fails to capture the whole contour but just part of it. In terms of optimization, every such solution corresponds to a *mediocre* local minimum since it intersects with some portion of the object's boundary owing to a crater-like weighting scheme (2). Thus the resulting level surface of f_e should be mostly



(a) Moving slowly (b) Moving fast (c) Synthesis #060 (d) Synthesis #154

Figure 3. Tracking with edge density information only.

smooth with a concentration of local minima near the objects's location at the previous frame. As to the global minimum, i.e., the one captures the whole contour, it could be either near-by when the object's motion is slow, or comparatively farther than many of the local minima when the motion is fast. For the second experiment, we create a synthesized image sequence of a monotone-color elliptic object undergoing various changes in its scale and orientation (without any translation). In this case an algorithm that tracks solely by comparing color distributions will fail to capture the right scale because of the uniform color of the target, i.e., the underlying tracking problem using only color distribution is ill-posed. Nevertheless, it is handled accurately by tracking with edge density, where we have shown the results for two image frames in Fig. 3c and Fig. 3d. The results and the analysis of the experiments justify the use of the proposed objective function defined in (6) by integrating both color distribution and edge information for tracking.

Overall, we have discussed the issues of object representation and search techniques for distribution-based tracking systems. By arguing a convergent local minimum should be not only *significant* but also *of interest*, we show that trust-region methods provide the techniques and controls for this aspect of consideration. This is also the main reason that we think trust-region is more appropriate for tracking than line-search. We have demonstrated this point by comparing with a well-known mean-shift tracker, via a number of experiments. Our other efforts have been made to design a good representation model. Since tracking by using color distribution only is not a well-posed problem, we consider a representation that integrates the object's geometry and its color distribution. While computing a target's characteristics within an area enclosed by a boundary of arbitrary shape is rather complicated, we contend with using the bivariate covariance ellipse to describe them. The resulting representation integrates color distribution and edge density information via two coupled weighting schemes, and it enables the system to perform optimization over a continuous space to yield more accurate results.

References

[1] S.T. Birchfield, "Elliptical Head Tracking Using Intensity Gradients and Color Histograms," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 232–237, Santa Barbara, CA, 1998.

[2] G.R. Bradski, "Computer Vision Face Tracking for Use in a Perceptual User Interface," *Intel Technology Journal*, 1998.

[3] H.T. Chen and T.L. Liu, "Trust-Region Methods for Real-Time Tracking," *Proc. Eighth IEEE Int'l Conf. Computer Vision*, vol. 2, pp. 717–722, Vancouver, Canada, 2001.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *Proc. Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 142–149, Hilton Head Island, South Carolina, 2000.

[5] A.R. Conn, N.I.M. Gould, and P.L. Toint, *Trust-Region Methods*, SIAM, 2000.

[6] Intel Corporation, *Intel Image Processing Library Reference Manual*, 2000, Document Number 663791-005.

[7] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-Based Probabilistic Tracking," *Proc. Seventh European Conf. Computer Vision*, vol. 1, p. 661 ff., Copenhagen, Denmark, 2002.

[8] C. Rasmussen and G.D. Hager, "Probabilistic Data Association Methods for Tracking Complex Visual Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 560–576, June 2001.

[9] T. Zhang and D. Freedman, "Tracking objects using density matching and shape priors," *Proc. Ninth IEEE Int'l Conf. Computer Vision*, pp. 1056–1062, Nice, France, 2003.