Trust-Region Methods for Real-Time Tracking

Hwann-Tzong Chen Tyng-Luh Liu
Institute of Information Science
Academia Sinica
Nankang, Taipei 115 Taiwan
{pras, liutyng}@iis.sinica.edu.tw

Abstract

Optimization methods based on iterative schemes can be divided into two classes: linesearch methods and trust-region methods. While linesearch techniques are commonly found in various vision applications, not much attention is paid to trust-region methods. Motivated by the fact that linesearch methods can be considered as special cases of trust-region methods, we propose to apply trust-region methods to visual tracking problems.

Our approach integrates trust-region methods with the Kullback Leibler distance to track a rigid or non-rigid object in real-time. If not limited by the speed of a camera, the algorithm can achieve frame rate above 60 fps. To justify our method, a variety of experiments/comparisons are carried out for the trust-region tracker and a linesearch-based mean-shift tracker with same initial conditions. The experimental results support our conjecture that a trust-region tracker should perform superiorly to a linesearch one.

1. Introduction

The primary goal of this work is to establish a real-time tracking framework using trust-region methods. We will describe the concepts of trust-region methods in detail, then illustrate the efficiency of the algorithm with examples.

Tracking methods based on *frame differencing* and *shape analysis* are most common. Pfinder [12] is a real-time system to perform one-person tracking using a multi-class statistical model of color and shape. Haritaoglu et al. [8] have proposed the the W^4 system to track people and their part structures in an outdoor environment. To deal with interactions among the tracked people, *appearance models* are used to resolve the ambiguities. In [11] a system for color image sequences is presented. The approach is similar to W^4 built upon a background model combining pixel RGB and chromaticity values with local image gradients.

Several systems have been proposed for 3D object tracking. In [7], Gavrila and Davis describe a decomposition approach and a best-first local search to reconstruct 3D pose with images taken from multiple views and predict the pose in the new coming image frames. Delamarre and Faugeras use 3D articulated models for human tracking with two or more cameras [5]. An interesting approach to recover moderate motions in image sequences is proposed by Bregler and Malik [2]. Their method is based on *twists* and *exponential maps* to recover high degree-of-freedom of motion configurations.

The I/CONDENSATION algorithms [9], [10] by Isard and Blake are used to track curves in clutter using stochastic analysis and *importance/factored sampling*. The methods are superior to previous Kalman filter based approaches, and achieves real-time performance.

Bradski has proposed a *Continuously Adaptive Mean Shift* (CAMSHIFT) algorithm for use in a perceptual user interface to track face [1]. The method is based on non-parametric technique and *mean shift* to find the peak mode of a color probability distribution. In [3], Comaniciu et al. apply mean shift to real-time tracking for non-rigid objects. They also model the objects by their color distributions, then measure the similarities between objects with the Bhattacharyya coefficient.

Our approach: Our system is an integration of *trust-region methods* and *Kullback Leibler distance*. To track a target, rigid or non-rigid, we compute its RGB color probability distribution, then use the Kullback Leibler distance to detect a similar distribution/object in each succeeding image frame. The Kullback Leibler distance is to measure the dissimilarity between the target distribution and a candidate distribution. In each image frame, to find out a distribution most similar to the target's in real time is not an easy task. Perhaps, an iterative optimization technique is the most appropriate way to do it. Judging from the efficiency and robustness of trust-region methods, we adopt them to perform the optimization processes. It turns out to be a perfect match for use in a real-time tracking system.

2. Trust-Region Methods

There are two classes of iterative algorithms for optimization: *linesearch methods* and *trust-region methods*. While linesearch techniques are commonly applied in various vision applications, not much attention is paid to trust-region methods by the computer vision community. In fact, linesearch methods can be considered as special cases of trust-region methods.

2.1. What Are Trust-Region Methods?

To explain the basic ideas of trust-region methods, we consider the following unconstrained optimization problem over an image frame I:

$$\min_{\mathbf{x}\in I} f(\mathbf{x})\,, \tag{1}$$

where $\mathbf{x} = [x_1, x_2]^T$ is any pixel in I, and $f(\mathbf{x})$ is some objective function to be minimized. Unlike the linesearch methods, trust-region methods will try to find the next approximate solution within a region of the current iterate.

Two issues need to be emphasized: (1) how to determine the size of the region, and (2) how to approximate a solution in the region. These turn out to be the two most important elements of any trust-region method. The first has to do with the *trust-region radius*, and the second is determined by solving a *trust-region subproblem*.

More precisely, suppose an initial point $\mathbf{x}_0$, and an initial trust-region radius $\triangle_0 > 0$ are given. Let η_1, η_2, γ_1, and γ_2 be some constants and satisfy

$$0 < \eta_1 \le \eta_2 < 1 \quad \text{and} \quad 0 < \gamma_1 \le \gamma_2 < 1.$$

Following [4], we use $\eta_1 = 0.01$, $\eta_2 = 0.9$, and $\gamma_1 = \gamma_2 = 0.5$. Then, at each iteration k with current iterate $\mathbf{x}_k$, and trust-region radius $\triangle_k$, the following two steps are performed until $\{\mathbf{x}_k\}$ converges.

1. Trust-region subproblem: A *model* m_k is constructed to approximate f within the trust region. A trust-region subproblem is then to compute an $\mathbf{s}_k$ such that m_k is "sufficiently reduced" and $\mathbf{x}_k + \mathbf{s}_k$ is in the region. In this work, we will use quadratic model to approximate the objective function, i.e.,

$$m_k(\mathbf{x}_k + \mathbf{s}) = m_k(\mathbf{x}_k) + < g_k, \mathbf{s} > + \frac{1}{2} < \mathbf{s}, H_k\mathbf{s} >\,,$$

where $m_k(\mathbf{x}_k) = f(\mathbf{x}_k)$, $g_k = \nabla_{\mathbf{x}} f(\mathbf{x}_k)$, and H_k is the Hessian. For visual tracking application, $H_k = \nabla_{\mathbf{xx}} m_k(\mathbf{x}_k)$ is a 2-by-2 symmetric matrix to approximate $\nabla_{\mathbf{xx}} f(\mathbf{x}_k)$. When $H_k \neq 0$, m_k is said to be a second-order model. Thus the corresponding trust-region subproblem is

$$\min_{\|\mathbf{s}\|\le\triangle_k} \psi_k(\mathbf{s}) = < g_k, \mathbf{s} > + \frac{1}{2} < \mathbf{s}, H_k\mathbf{s} > . \tag{2}$$

2. Trust-region radius: After solving the subproblem, the trial point $\mathbf{x}_k + \mathbf{s}_k$ will be tested to see if it is a good candidate for the next iterate. This is evaluated explicitly by the following formula:

$$\rho_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{s}_k)}{m(\mathbf{x}_k) - m(\mathbf{x}_k + \mathbf{s}_k)}\,, \tag{3}$$

where if $\rho_k \ge \eta_1$, then the trial point is accepted, i.e., $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$. Otherwise, $\mathbf{x}_{k+1} = \mathbf{x}_k$. Since η_1 is a small positive number, the above rule adopts the trial point only when the value of objective function f is also reduced. When m_k approximates f well and yields a large ρ_k, the trust-region radius will be expanded for the next iteration. On the other hand, if ρ_k is smaller than η_1 or is negative, it suggests that the objective function f is not well approximated by the model function m_k within the current trust region. Therefore, the trust-region radius will be reduced to derive a more appropriate subproblem for the next iteration. The new trust-region radius can be updated as follows.

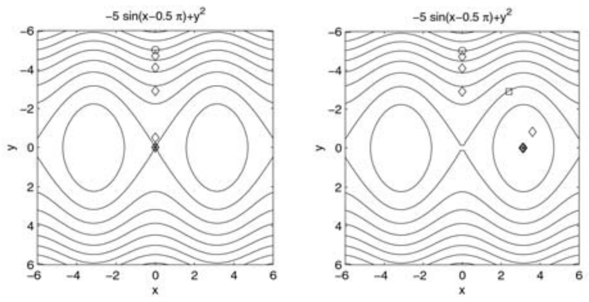
$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho_k \ge \eta_2, \\ [\gamma_2\Delta_k, \Delta_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_1\Delta_k, \gamma_2\Delta_k] & \text{if } \rho_k < \eta_1. \end{cases}$$

2.2. Trust-Region vs. Linesearch

Since linesearch methods are indeed special cases of trust-region methods, there are some favorable properties of trust-region methods not shared by the linesearch ones. For example, during an optimizing process, trust-region radiuses are adjusted adaptively to the shape of the objective function so that it can have better approximations. Also, unlike a linesearch method that optimizes only along the *descent* direction, trust-region methods carry out an optimization task based on region information. In the following, we describe two schemes that can be incorporated into trust-region methods for further improvements.

Non-convex subproblem: Due to the trust region constraint, a second order trust-region subproblem may be non-convex, i.e., H_k may not be positive semidefinite. When this happens, it may cause the trust-region iterates to be trapped into a saddle point. To avoid this, observe that the least eigenvalue of a non-convex H_k must be negative. The negative curvature information provides the other possible candidate when computing the next iterate. In [4], such point is named as *eigenpoint*, it is the intersection of the trust-region boundary and a straight line along the direction of the corresponding eigenvector for the negative eigenvalue. To illustrate, consider the following example. We use a linesearch gradient-descent method and a trust-region method to minimize the objective function $f(x, y) = -5\sin(x - 0.5\pi) + y^2$. Both start from the same point $(0, -4.7)$, but the linesearch one will end up at the saddle point $(0, 0)$, and the trust-region one will converge to the right answer (see Figure 1(a),(b)).

Non-monotone convergence: In general, during a minimization process, the sequence of $\{f(\mathbf{x}_k)\}$ at the iterates $\mathbf{x}_k$ is monotonically decreasing. For trust-region methods, it is possible to relax the monotone condition so that the algorithms are more flexible and efficient. This is achieved by defining a *reference index* $r(k)$ for each iteration k such that $r(0) = 0$ and $r(k) < k$. Then, instead of imposing the monotonicity condition on $\{f(\mathbf{x}_k)\}$, we now require $f(\mathbf{x}_{r(k-1)}) < f(\mathbf{x}_{r(k)})$.



(a) Gradient descent (b) Trust region

Figure 1. (a) A linesearch scheme based on gradient descent is trapped into a saddle point, $(0, 0)$. (b) Using negative curvature information, a trust-region method can locate the eigenpoint, marked as a square, and find its way to one of the minima.

3. Tracking with Trust-Region Methods

In this section, we explain how to apply trust-region methods for real-time visual tracking of single target object.

3.1. Objective Function for Tracking

Motivated by [1] and [3], our tracking system uses color distributions to represent objects. The RGB color space is divided into n bins, and a well-defined single-valued bin assignment function b is defined uniquely by pixel's RGB value as $b : \mathbf{y}_i \mapsto \{1, \ldots, n\}$, where $\mathbf{y}_i$ is any pixel in an image. An object can then be represented by its RGB color distribution using a color histogram analysis.

To begin with, we need to define the probability distribution for the target object to be tracked. Let I^0 be the first image frame, and $A(\mathbf{x})$ denote a square area centered at pixel $\mathbf{x}$. Suppose a square area $A(\mathbf{x}^0)$ is chosen from I^0 to be the target object. Then, its color probability distribution, denoted as $p(u; \mathbf{x}^0)$, is defined by

$$p(u; \mathbf{x}^0) = \frac{1}{C_p} \sum_{\mathbf{y}_i \in A(\mathbf{x}^0)} w_i(\mathbf{y}_i; \mathbf{x}^0)\delta(b(\mathbf{y}_i) - u)\,, \qquad (4)$$

where w_i is a weighting function to adjust the contribution of $\mathbf{y}_i$ in the probability distribution, and δ is the Kronecker delta function. Notice that the constant C_p is the total weight, i.e., $C_p = \sum_{\mathbf{y}_i \in A(\mathbf{x}^0)} w_i(\mathbf{y}_i; \mathbf{x}^0)$. This implies $\sum_{u=1}^{n} p(u; \mathbf{x}^0) = 1$. The notation $p(u; \mathbf{x}^0)$ will be simplify to $p(u)$ if the omission does not cause any confusion.

Similarly, during tracking, an area enclosing by $A(\mathbf{x})$ at the tth image frame I^t, its color probability distribution, denoted as $q(u; \mathbf{x})$, is

$$q(u; \mathbf{x}) = \frac{1}{C_q} \sum_{\mathbf{y}_i \in A(\mathbf{x})} w_i(\mathbf{y}_i; \mathbf{x})\delta(b(\mathbf{y}_i) - u)\,, \qquad (5)$$

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

With the above probability distribution models, the problem of tracking the target object can be transformed into finding a color distribution $q(u; \mathbf{x})$ most similar to $p(u)$ at each image frame. In our system, the *Kullback Leibler distance* is used to measure the similarity between two probability distributions. Given two distributions $p(u)$ and $q(u; \mathbf{x})$, their Kullback Leibler distance is denoted as $D(p(u)||q(u; \mathbf{x}))$.

We are now in a position to define the objective function f used for tracking with trust-region methods. At each image frame I^t, the objective function f is defined by

$$f(\mathbf{x}) = D(p(u)||q(u; \mathbf{x})) = \sum_{u=1}^{n} p(u) \log \frac{p(u)}{q(u; \mathbf{x})}\,, \qquad (6)$$

where pixel $\mathbf{x} \in I^t$ is a candidate position which the target object may located at.

3.2. Model Approximation and Analysis

At image frame I^t, the trust-region methods can be applied to detect where the target is most likely located by minimizing $f(\mathbf{x})$. Starting from $\mathbf{x}^{t-1}$, the optimizer will try to find a convergent sequence of points by solving the trust-region subproblems iteratively. The final convergent point will be the new target location, i.e., $\mathbf{x}^t$.

Two issues need to be addressed when considering a trust-region subproblem. The first is how to construct a model appropriately approximating the objective function in the trust region; the second is how to solve the underlying subproblem with the chosen model.

We adopt a second order model m_k to approximate the Kullback Leibler objective function f, i.e., at each iteration k, the corresponding subproblem (2) will have

$$\begin{aligned} g_k &= \nabla_{\mathbf{x}} f(\mathbf{x}_k) = \nabla_{\mathbf{x}} \left(\sum_{u=1}^{n} p(u) \log \frac{p(u)}{q(u;\mathbf{x}_k)} \right), \\ H_k &= \nabla_{\mathbf{xx}} f(\mathbf{x}_k) = \nabla_{\mathbf{xx}} \left(\sum_{u=1}^{n} p(u) \log \frac{p(u)}{q(u;\mathbf{x}_k)} \right). \end{aligned}$$

To solve the subproblem, we use the *conjugate gradient method* to approximate an $\mathbf{s}_k$ sufficiently reducing the model. There are three possible cases:

- If H_k is positive definite and the minimum of $\psi_k(\mathbf{s})$ in (2) is inside the trust region, then the $\mathbf{s}_k$ is set to the minimizer.
- If H_k is positive definite and the minimum is outside the trust region, $\mathbf{s}_k$ will be set to the intersection between the negative gradient $-g_k$ and the trust region boundary.
- If H_k has a negative eigenvalue, then $\mathbf{s}_k$ will be set to either the the intersection between $-g_k$ and the trust region boundary or the eigenpoint, depending on which one reduces the model most.

3.3. Trust Region vs. Mean Shift

To justify our approach, we compare the performances of a trust-region tracker with a mean-shift tracker [3]. The main reason is that a tracker based on mean shift is driven solely by a linesearch engine along a descent direction. So, from our analysis in the previous sections, a tracking system using trust-region methods is expected to perform better.

In [3], the Bhattacharyya coefficient, defined by $f(\mathbf{x}) = \sum_{u=1}^{n} \sqrt{p(u)q(u;\mathbf{x})}$, is used as the objective function to be maximized for tracking. In order to carry out a careful and fair comparison, we have implemented the mean-shift tracker exactly following the steps described in that paper. The experimental results for the two methods will be discussed later. For now, let's focus on what makes the two approaches different.

Firstly, we have noticed that a Bhattacharyya objective function typically results in a smoother level surface for each image frame, compared to the one produced by the Kullback Leibler distance (see Figure 2 (a),(b)).

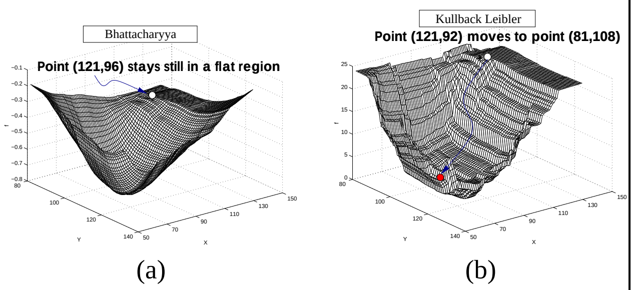


Figure 2. (a) The Bhattacharyya level surface of frame 118 of *car* sequence. (b) The Kullback Leibler level surface of frame 118 of *car* sequence (see Figure 4(k)).

For a further investigation, we have tested trust-region methods with a Bhattacharyya coefficient. The performance is similar to a mean-shift tracker. Then, the Kullback Leibler distance is used as the objective function for a mean-shift tracker. However, such combination has degraded the mean-shift tracker's performance a lot. Using Figure 3(a) as an example, the two points, 1 and 4, inside the small square are the car's positions at previous and current frame, respectively. For a tracker to track the target correctly, its optimizer should be able to move from point 1 to point 4. In Figure 3(c), the Kullback Leibler level curves corresponding to the small square area are plotted, where an enlarge portion is plotted in Figure 3(d). When using a trust-region method, it takes 3 iterations to converge to point 4 ($1 \to 2 \to 3 \to 4$). Nevertheless, a mean-shift tracker will first try to reach point A (see Figure 3(d)), then find out the energy there is higher so that it will iterate backward along the line passing through 1 and A. In this case all the backward iterates happen to have higher level values. This causes the mean-shift tracker to stay at point 1, and misses the target.Taken all the above discussion into account, we have made the following observations.

- Since a trust-region method can adapt to the shape of an objective function by dynamically adjusting the trust-region radiuses, it is more general and robust to accommodate various objective functions. This also reflects in the tracker's speed. A trust-region tracker is much faster than a linesearch one. It can have large steps between iterates when the trust-region radius is large. Contrarily, each step of a mean-shift tracker is limited by the size of search widow.
- The mean shift method is limited by the linesearch nature so that it is sensitive to the selection of an objec-

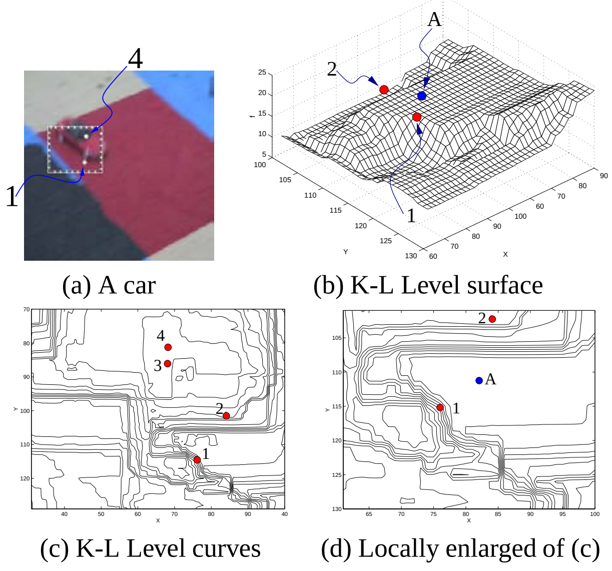


Figure 3. The Kullback Leibler level surface/curves and tracking outcomes of an image frame from *car* sequence.

tive function. In fact, our experimental results suggest that the Kullback Leibler distance is a better choice for use in tracking. However, it does not match well with mean shift since it will yield rather non-smooth level surfaces when used as an objective function.

4. Experimental Results and Discussion

We have presented a new tracking algorithm based on trust-region methods. Depending on the target size, the system runs efficiently with a frame rate above 60 fps on a P-III 733 PC. A variety of experiments have been carried out to test our method. In each test, the RGB space is divided into $16 \times 16 \times 16 = 4096$ bins and a Gaussian weighting scheme is adopted for w_i. For comparison, each video sequence is also tested separately with a mean-shift tracker using the same initial conditions. The results we obtain are encouraging, and they support our speculation that a trust-region based tracker should outperform a linesearch-based tracker. Some of the test results are shown in Figure 4, where the frame rates for each test are listed in Table 1. For the future work, we are now working on extending the trust-region tracker to handle scale and orientation changes, and to track multiple objects simultaneously.

Method	Trust Region	Mean Shift
Amy $A(\mathbf{x}_0)$ Size: 34x34	73.05 fps	58.28 fps
Car $A(\mathbf{x}_0)$ Size: 42x42	53.73 fps	35.81 fps
Magnet $A(\mathbf{x}_0)$ Size: 30x30	78.54 fps	51.41 fps

Table 1. Frame rates for a trust-region tracker and a mean-shift tracker, respectively.

Acknowledgments

This work was supported in part by an NSC grant 89-2218-E-001-010.

References

[1] G. R. Bradski, “Computer Vision Face Tracking For Use in a Perceptual User Interface,” *Intel Technology Journal*, Q2, 1998.

[2] C. Bregler and J. Malik, “Tracking People with Twists and Exponential Maps,” *CVPR98*, pp.8-15, Santa Barbara, CA.

[3] D. Comaniciu, V. Ramesh and P. Meer, “Real-Time Tracking of Non-Rigid Objects using Mean Shift,” *CVPR*, Vol. 2, pp. 142-149, Hilton Head, SC, 2000.

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

[5] Q. Delamarre and O. D. Faugeras, “3D Articulated Models and Multi-View Tracking with Silhouettes,” *ICCV99*, pp. 716-721, Greece.

[6] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Second Ed., Academic Press, Boston, 1990.

[7] D. M. Gavrila and L.S. Davis, “3D Model-Based Tracking of Humans in Action: A Multi-View Approach,” *CVPR*, pp. 73-80, San Francisco, CA, 1996.

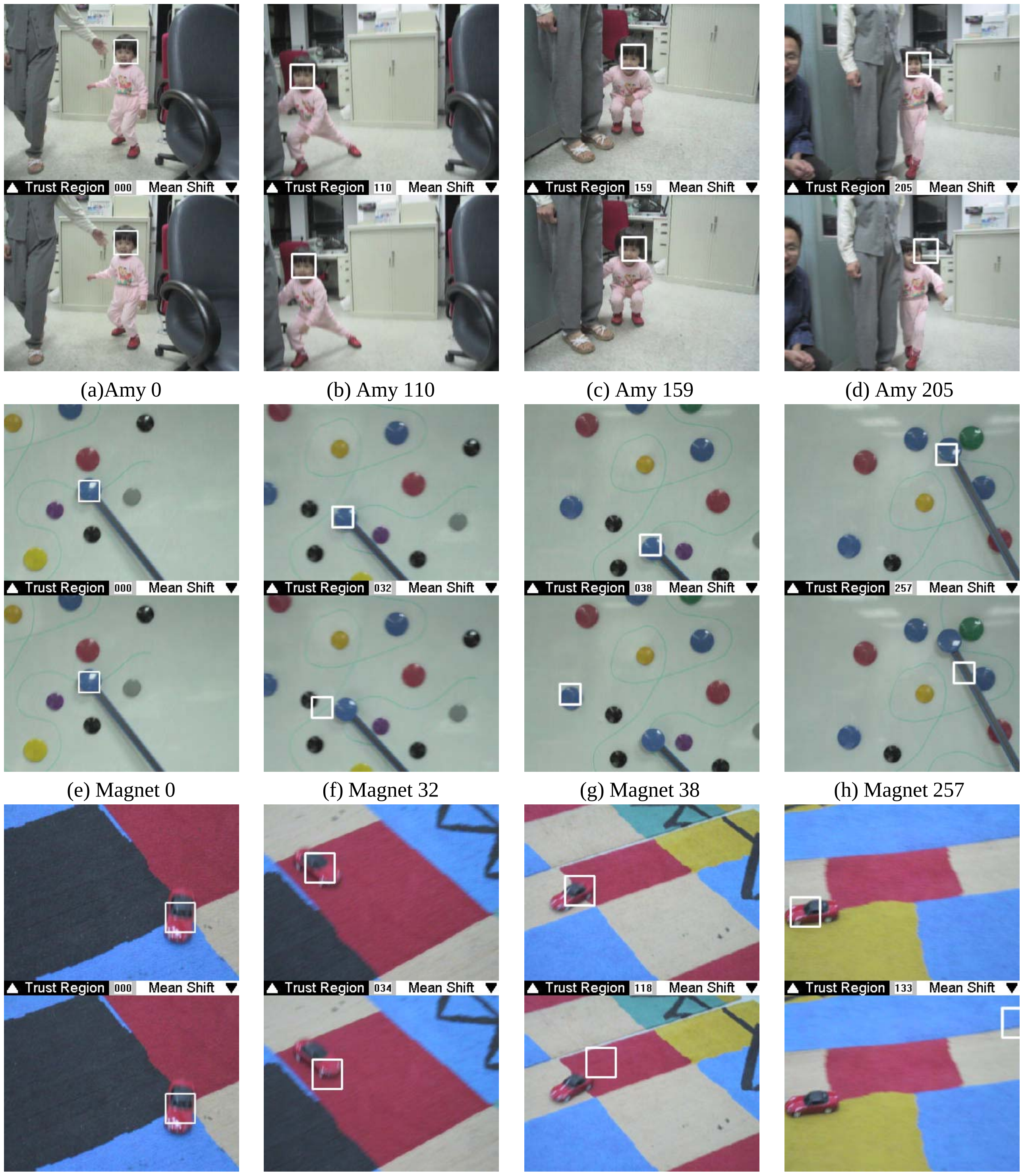
[8] I. Haritaoglu, D. Harwood, and L. Davis, “W4: Who? When? Where? What? A Real Time System for Detecting and Tracking People,” *AFGR*, Nara, Japan, 1998.

[9] M. Isard and A. Blake, “Contour Tracking by Stochastic Propagation of Conditional Density,” *ECCV96*, pp. 343-356, England.

[10] M. Isard and A. Blake, “CONDENSATION – Conditional Density Propagation for Visual Tracking,” *Int. J. Computer Vision*, (29), 1, pp. 5-28, 1998.

[11] S. J. McKenna, S. Jabri, Z. Duric and H. Wechsler, “Tracking Interacting People,” *AFGR*, Grenoble, France, 2000.

[12] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland, “Pfinder: Real-Time Tracking of the Human Body,” *PAMI*, 19(7), pp. 780-785 July 1997.


Trust Region 000 Mean Shift
Trust Region 110 Mean Shift
Trust Region 159 Mean Shift
Trust Region 205 Mean Shift
Trust Region 000 Mean Shift
Trust Region 032 Mean Shift
Trust Region 038 Mean Shift
Trust Region 257 Mean Shift
Trust Region 000 Mean Shift
Trust Region 034 Mean Shift
Trust Region 118 Mean Shift
Trust Region 133 Mean Shift

(a)Amy 0 (b) Amy 110 (c) Amy 159 (d) Amy 205

(e) Magnet 0 (f) Magnet 32 (g) Magnet 38 (h) Magnet 257

(i) Car 0 (j) Car 34 (k) Car 118 (l) Car 133

Figure 4. (a)-(d) *Amy* **sequence. The two trackers perform almost the same since there is not much distraction in the background. (e)-(f)** *Magnet* **sequence. The mean-shift tracker starts to miss the target at the 32nd frame (the number between images is the frame number). (i)-(l)** *Car* **sequence. The mean-shift tracker loses the target due to the rapid motion and similar background color.**