



HeadTalk

Understanding the Nature and Role of Gesture in One-to-One Conversation

ESRC Small Grants Project	RES 149 25 1016
Document ID	HeadTalk Deliverable
Status	Draft
Type	HeadTalk Demonstrator: Supporting Documentation
Version	0.1
Date	December 16th 2006
Editors	Asad Naeem, Tony Pridmore, and Steven Mills

© ESRC Small Grants Project RES 149 25 1016 HeadTalk

Project Co-ordinators:

Ronald Carter
School of English Studies
University of Nottingham
Trent Building
University Park
Nottingham
NG7 2RD
United Kingdom

Andy Crabtree
School of Computer Science & IT
University of Nottingham
Jubilee Campus
Wollaton Road
Nottingham
NG8 1BB
United Kingdom

Tel. +44 (0)115 95 15902
Fax. +44 (0)115 95 15924
Email. ronald.carter@nottingham.ac.uk

Tel. +44 (0)115 84 66512
Fax. +44 (0)115 84 66416
Email. andy.crabtree@nottingham.ac.uk

Editors of this report:

Asad Naeem, Tony Pridmore and Steven Mills

School of Computer Science & IT, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom.



Table of Contents

1. Overview.....	1
2. Installation and Use.....	1
3. Future Work.....	5
4. References.....	5
Appendix 1. Managing Particle Spread with a Hybrid Particle Filter/Kernel Mean-Shift Tracker.....	6

1. Overview

The HeadTalk demonstrator, Cvision, is an interactive program which allows users to apply the visual tracking algorithm developed within the HeadTalk project [1] to selected targets in an input video clip. This algorithm, KAMS (Kernel Annealed Mean-Shift tracking), is a hybrid particle filter/kernel mean-shift tracker formed by combination of the Annealed Particle Filter of Deutscher et al. [2] and the Kernel Mean-Shift algorithm of Comaniciu et al. [3]. Though KAMS has been shown to outperform the original particle filter of Isard and Blake [4], Annealed Particle Filtering [2] and the earlier hybrid tracker of Maggio and Cavallero [5], Cvision also incorporates, for comparison, each of these three algorithms.

Cvision takes as its input an avi format video file and produces a text file giving the estimated position of each target in each frame of that video. This may be imported into the current version of the DreSS 2 tool DRS [6]. An output video may also be produced if desired, this shows the results of tracking overlaid on the input video images and is a useful debugging and interpretation tool. Cvision allows multiple targets to be tracked in parallel, producing a description of the motion of each and showing intermediate results as they are obtained. Cvision is written in C++ and provided as a Windows .exe file.

2. Installation and Use

Cvision is available from the NCESS website (URL?) as a zip file `cvision.zip`. To install the program simply open `cvision.zip` with Winzip, Powerarchiver or similar and save the executable `cvision.exe` to a suitable location on your machine. Cvision will run on most PCs, but was written under Windows XP, and may need recompiling for some systems. Should you experience problems running the program, please contact Dr. Tony Pridmore at tpp@cs.nott.ac.uk (Please note, the Cvision source code is not being released at time of writing).

Running `cvision.exe` will bring up the simple GUI shown in Figure 1. The radio buttons to the left hand side allow the user to select the algorithm to be applied. *Condensation* invokes the original particle filter of Isard and Blake [], *Hybrid* invokes the Condensation/Mean Shift hybrid tracker of Maggio and Caravello [] and *Annealed Filter* chooses the annealed particle filter of Deutscher et al []. To select the HeadTalk tracker, simply left click on the radio button marked *KAMS*.

Each algorithm makes use, in a variety of ways, of a set of discrete hypotheses, termed particles, of the likely position of the target object. The number of particles employed can be set by the user in the *Tracking Parameters* section to the right of the GUI. Increasing the number of particles makes it more likely that the target will be tracked correctly, as a wider set of hypotheses will be considered, but adds computational overhead. The default value of 100

particles has been set after experimentation with a large number of video clips and is a good starting point for exploration.

The Cvision algorithms all use the same representation of the target object; a three-dimensional colour histogram, computed over the RGB colour space. Once the user has selected a target region, this histogram is computed and matched to subsequent frames to identify the latest position of the target. The second element of the *Tracking Parameters* section allows the user to determine the number of bins in the histogram. Increasing this value leads to a more accurate model of the target object, as seen in the first frame of the video, being formed. Accuracy is attractive. The more accurate the object representation, the less likely the tracker is to confuse that object with another, similar item. Accuracy, however, also brings a potential risk. If the histogram is too fine-grained, the system will seek a very precisely defined object in subsequent frames. Image noise is never completely removed from real video records and may distort the true target enough to make it appear dissimilar to the original histogram. A compromise is needed between a histogram that is detailed enough (has a large enough number of bins) to capture the range of colours in the object, but coarse enough to withstand the inevitable changes in colour brought about by image noise.

The final element of the Tracking Parameters section of the Cvision interface is a check box labelled *Store Video Tracking Results*. A left click here will cause the system to produce a video file showing the chosen tracker's estimate(s) of target position(s) overlaid on each frame of the input video.

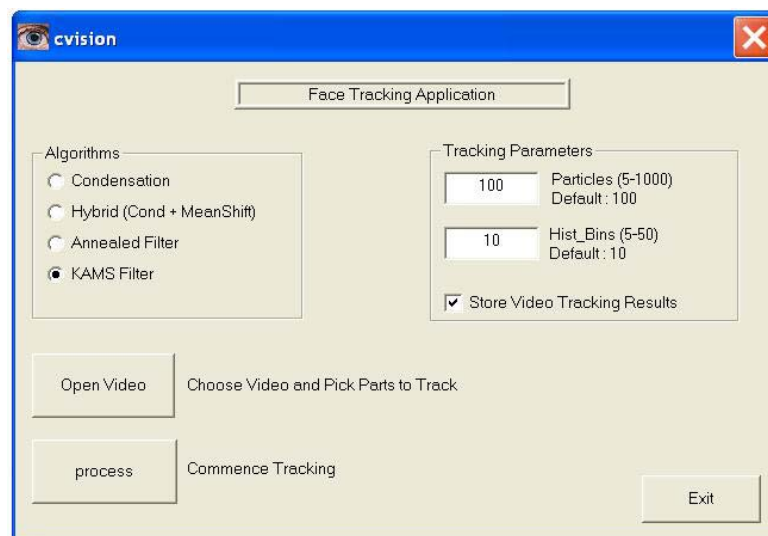


Figure 1. The main Cvision interface

To apply Cvision:

- start up the program by double clicking on the Cvision icon
- select an algorithm using the *Algorithms* radio buttons
- set the Tracking Parameters
 - No. of particles

- o No. of bins in the colour histogram
 - o Whether or not to store a video of the result
- Click on *Open Video* to access the input video. This brings up a dialog box which allows you to select an avi file for tracking (Figure 2). Select a file in the usual way.

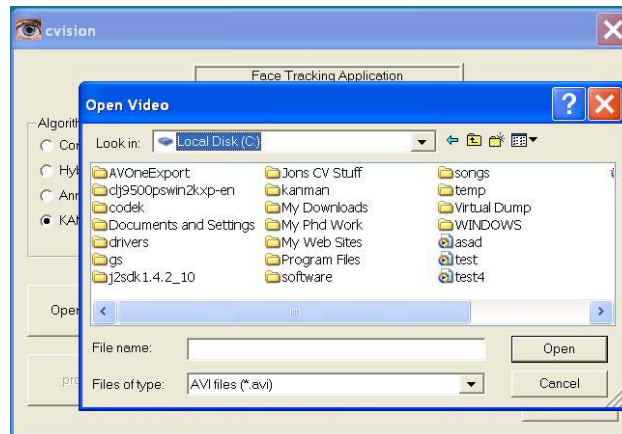


Figure 2. Selecting an input video.

- When the video file is open a second window will appear (Figure 3) labelled *Pick Area to Track* and showing the first frame of the input sequence.

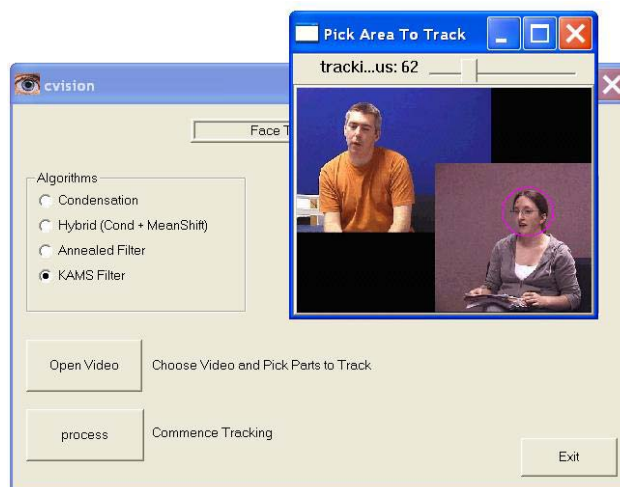


Figure 3. Specifying targets

- Left clicking anywhere on the image will initialise a target to be tracked. It may take more than one attempt to correctly select the desired target, so each left click erases the previous target and inserts a new one. The slider at the top of the window changes the radius of the target area. In figure 3 a target of radius 62 pixels is centred on the female participant's head.

- To select multiple targets right click after each selection. Targets already selected will then remain in place while new ones are added.
- When all the required targets are selected, left click on *process* in the main interface to begin tracking. As tracking proceeds, each frame of the video will be displayed, with the circle(s) marking the target(s) overlaid on each frame. When the end of the sequence is reached the *Pick Area to Track* window will close.
- The results of tracking are written to a file named *output.txt* in the same directory as the input video. This file contains an initial line stating the number of tracked objects, a row of asterisks to separate this from the main data block, and then the image coordinates of each target (in the order in which they were selected) in each frame. Figure 4 shows a sample *output.txt* file containing the result of tracking two targets through a 20 frame sequence.

```

Number Of Tracked Objects = 2
*****
frame  X-cord  Yc-cord  frame  X-cord  Yc-cord
  1     87     86     1     35     247
  2     84     89     2     35     251
  3     86     89     3     31     253
  4     82     88     4     36     252
  5     84     92     5     32     248
  6     82     89     6     28     251
  7     87     89     7     28     254
  8     81     88     8     31     253
  9     82     88     9     35     251
 10     83     87    10     34     252
 11     83     91    11     28     251
 12     83     91    12     32     251
 13     82     88    13     30     252
 14     84     88    14     30     248
 15     82     90    15     30     250
 16     84     87    16     32     250
 17     80     92    17     34     246
 18     81     88    18     31     250
 19     83     89    19     35     249
 20     82     88    20     28     251

```

Figure 4. Sample output.txt file

- If the *Store Video Tracking Results* option is checked, a dialog box will appear when tracking commences which allows the compression applied to the output video to be selected (Figure 5). A video named *result.avi* will then appear in the same directory as the input video showing the output of the tracker overlaid on each frame of the input image. The output comprises circles showing the best estimate of the target position, and a set of purple dots. Each dot shows one particle, and allow the range of possibilities considered by the tracker to be evaluated.

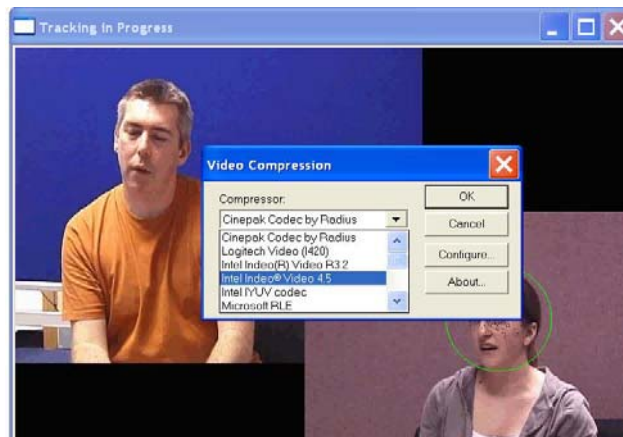


Figure 5. Selecting a video compression method.

3. Future Developments

It should be stressed that the current version of Cvision is beta-test only. Though the core tracking algorithms are well-developed the GUI is less so and a number of extensions and improvements are required to make the program easier to use and fully robust. At present the major known limitations are:

- The program is designed to process one video, as described above, each time it is run. Although Cvision will allow users to interact more flexibly with it this often causes the program to crash.
- Only some avi compression styles are supported.
- There is no facility to rename output files.

Future releases will address these issues.

4. References

- [1] A. Naeem, T. Pridmore and S. Mills, Managing Particle Spread with a Hybrid Particle Filter/Kernel Mean-Shift Tracker, paper submitted on Dec. 3rd 2006 to IEEE International Conference on Computer Vision and Pattern Recognition, Minneapolis, Minnesota, 2007. **(Appendix 1)**
- [2] J. Deutscher, A. Blake, and I. Reid, Articulated body motion capture by annealed particle filtering, Proc. IEEE Conf. Computer Vision Pattern Recognition, 2000.
- [3] D. Comaniciu, V. Ramesh, and P. Meer, Kernel-based object tracking, IEEE Trans. Pattern Analysis and Machine Intelligence, 25(5), pp. 564–577, 2003.
- [4] M. Isard and A. Blake, CONDENSATION – conditional density propagation for visual tracking, International. Journal of Computer Vision, 29(1) pp5-28, 1998.
- [5] E. Maggio and A. Cavallaro, Hybrid particle filter and Mean Shift tracker with adaptive transition model, Proc. Int. Conf. Acoustics, Speech, and Signal Processing 2005.
- [6] A. French, C.Greenhalgh, A. Crabtree, M. Wright, P. Brundell, A. Hampshire and T. Rodden, [Software Replay Tools for Time-based Social Science Data](#). Paper delivered at the *2nd annual international e-Social Science conference*, June 2006, University of Manchester, 2006.

Appendix 1: Managing Particle Spread with a Hybrid Particle Filter/Kernel Mean-Shift Tracker

Paper submitted on Dec. 3rd 2006 to the IEEE International Conference on Computer Vision and Pattern Recognition, Minneapolis, Minnesota, 2007.

Abstract

Particle filtering provides a well-developed and widely adopted approach to visual tracking. For effective tracking in real-world environments the particle set must sample widely enough that it can represent alternative target states in areas of ambiguity. It must not, however, become diffuse, spreading across the image plane rather than clustering around the object of interest. A key issue in the design of particle filter-based trackers is how to manage the spread of the particle set to balance these conflicting requirements. To be computationally efficient, balance must be achieved with as small a particle set as reasonably possible. A number of hybrid particle filter/mean-shift trackers have recently been proposed which alternately disperse and cluster particles together, providing both a measure of balance and a reduced particle set. We present a novel hybrid of the annealed particle filter and kernel mean-shift algorithms. The proposed algorithm has been applied to artificial and real image sequences under a wide variety of conditions. Examination of the results shows the method to have both performance and efficiency advantages over existing particle filter/kernel mean-shift hybrid trackers.

1. Introduction

Visual tracking has received much attention in recent years, with template matching [1] Kalman and particle filtering [2] being among the most successful and widely adopted general approaches. Each approach has its strengths and weaknesses. The Kalman filter, for example, has a sound theoretical basis, and is effective and efficient in operation. Kalman filters, however, assume a linear motion model and a unimodal, and Gaussian, posterior.

The strength of the particle filter lies in its use of a set of discrete particles to represent multi-modal probability distributions that capture and maintain multiple hypotheses about target properties. Particle filtering is an iterative process in which particles are repeatedly selected, projected forwards using a motion model, dispersed by an additive random component, and evaluated against the image data. Many particle filter-based trackers have been developed since Blake and Isard first introduced the Condensation algorithm [2].

The ability of a set of particles to represent a wide variety of distributions is both the main strength and primary weakness of the particle filter. For effective tracking in real-world environments the particle set must sample widely enough to represent all possible

alternatives in areas of ambiguity. It must not, however, become diffuse, spreading across the image plane rather than clustering around the object of interest. When this happens particles tend to migrate towards local maxima in their evaluation function, becoming caught on clutter and losing track of the target. Similarly, particles should not become too focused. Though it is encouraging to see a particle set coalesce when a single, clearly distinguishable target moves across the image, the tracker should not become locked onto a single mode as does the Kalman filter.

A key issue in the design of particle filter-based trackers is how to manage the spread of the particle set to balance these conflicting requirements. The standard deviation of the posterior is simply and elegantly maintained by the Kalman filter, but particle filters cannot assume a Gaussian, or indeed any specific, distribution. Moreover, balance must be achieved with as small a particle set as reasonably possible – increasing the particle set increases the accuracy with which it represents the posterior, but adds significantly to computational overhead.

Several works have addressed aspects of this problem. Some point out that, in practice, the advantages of the particle filter approach are often lost as particles cluster, sometimes very quickly, around one target hypothesis. They focus on maintaining a wider distribution. The Annealed Particle Filter [3] uses annealing to smooth out the evaluation function, making the global maximum clearer and allowing particles to be spread further, by increasing the process noise, without becoming caught on local clutter. Vermaak et al [4] explicitly model the particle distribution as a Gaussian mixture model, forcing the resulting filter to sample an equal number of particles from each model component. This prevents a single, slightly more highly weighted, mode from dominating the particle distribution.

Other workers consider standard algorithms to spread the particle set too thinly across the image and concentrate their efforts on forcing particles to coalesce, reducing the number needed, and so computational expense. The Kernel Particle Filter [5] applies the Mean Shift hill-climbing algorithm to the particle set to pull the centre of the particle distribution towards the target centre. The Kernel Particle Filter can be effective, but clusters weighted particles without further reference to the image data.

More recently, Maggio and Cavallaro [6] used the Kernel Mean Shift tracking algorithm [1] to move particles towards local maxima of the evaluation function on each iteration of Condensation [2], preventing the particle set from dispersing and reducing the number of particles required. Kernel Mean Shift tracking is a hill climbing approach which first computes the likelihood of each pixel in a circular search space around the prior target centre being the next target centre, then moves the previous centre towards the maximum likelihood solution. The effect is similar to the Kernel Particle Filter [5] but takes local image properties into account.

Though the authors focus on the computational savings made, Maggio and Cavallaro's [6] tracker can be viewed as attempting to manage particle spread by alternately diffusing the particle set using Condensation and clustering them with kernel Mean Shift. The algorithm

shows performance advantages over both Condensation and Mean Shift tracking, but has some drawbacks. If Condensation tends towards an incorrect local maximum the mean-shift step will accelerate the process.

Recognising that the weakness of the hybrid Condensation/Mean Shift tracker lies in the particle set generated by the Condensation component, Naeem et al [7] propose an alternative hybrid in which Kernel Mean Shift is the dominant technology. A small number of particles are generated, in a structured fashion, to explore further when confidence in the Mean Shift algorithm becomes low. Naeem et al's tracker makes explicit the iterative diffuse-cluster structure implicit in Maggio and Cavallaro's hybrid, and shows performance advantages over Condensation, Kernel Mean-Shift and the Maggio and Cavallaro hybrid. The method, however, requires the user to specify both the conditions under which extra particles are spawned and the (fixed) size of the region to be searched, which is irksome and open to human error.

We take an alternative approach. Rather than shift control away from the particle filter component and towards the kernel mean-shift tracker we replace Condensation with a more powerful particle filter. We propose a hybrid particle filter/mean-shift tracking algorithm created by combination of the kernel Mean-Shift algorithm with Deutscher et. al.'s [3] Annealed Particle filter. We hypothesise that by smoothing out local maxima in the evaluation function the annealed particle filter will allow a greater spread in the particle set, while the Mean-Shift component will successfully pull particles back towards the true target.

The proposed Kernel Annealed Mean Shift (KAMS) tracking algorithm is presented in Section 2. The KAMS filter has been implemented and is experimentally compared with Maggio and Caravello's [6] Hybrid and Naeem et al's [7] SOK filters in Section 3. Finally, conclusions are drawn in Section 4.

2. The Kernel Annealed Mean Shift Tracker

Annealed particle filtering relies upon a series of particle weighting functions $w_0(\mathbf{Z}, \mathbf{X})$ to $w_M(\mathbf{Z}, \mathbf{X})$ where \mathbf{Z} is a measurement vector extracted from the image and \mathbf{X} is the current model state. A given weighting function w_m is obtained by raising the original weighting function $w(\mathbf{Z}, \mathbf{X})$ to a power β_m , so that

$$w_m(\mathbf{Z}, \mathbf{X}) = w(\mathbf{Z}, \mathbf{X})^{\beta_m} \quad (1)$$

where $\beta_0 = 1.0$ and $\beta_0 > \beta_1 > \beta_2 > \dots > \beta_M$. As β_m increases, extrema in the weighting function become more pronounced. So $w_0(\mathbf{Z}, \mathbf{X})$ is the raw weighting function while $w_M(\mathbf{Z}, \mathbf{X})$ captures only the broad structure of the search space. In [3] $w(\mathbf{Z}, \mathbf{X})$ is the sum of squared differences between the model and image data.

In Annealed Particle Filtering each particle is evaluated at each time step using each $w_m(\mathbf{Z}, \mathbf{X})$, starting with $w_M(\mathbf{Z}, \mathbf{X})$ and moving to $w_0(\mathbf{Z}, \mathbf{X})$. At a given time step t_k the process begins with a set of N unweighted particles

$$S_{k,M} = \{s_{k,M}^{(0)}, s_{k,M}^{(1)}, \dots, s_{k,M}^{(N)}\} \quad (2)$$

Each particle $s_{k,M}^{(i)}$ is then assigned a weight $\pi_{k,m}^{(i)}$ where

$$\pi_{k,m}^{(i)} \propto w_m(\mathbf{Z}_k, s_{k,m}^{(i)}) \quad (3)$$

and, in the first step, $w_m(\mathbf{Z}_k, s_k^{(i)}) = w_M(\mathbf{Z}_k, s_k^{(i)})$, resulting in a set of weighted particles $S_{k,M}^\pi$. N particles are now drawn randomly from $S_{k,M}^\pi$ with replacement and used to create a set of unweighted particles for evaluation using the next weighting function

$$s_{k,M-1}^{(i)} = s_{k,M}^{(i)} + B_m \quad (4)$$

where B_m is a multi-variate Gaussian random variable with mean 0 and variance P_m . $s_{k,M-1}$ is then weighted using $w_{m-1}(\mathbf{Z}_k, s_k^{(i)})$. This process is repeated until $S_{k,0}^\pi$ is produced

Annealing allows us to counteract the natural tendency of particle filters to cluster particles together by increasing the variance P_m , confident that the smoother weighting functions used in the early part of the annealing run will steer particles away from local extrema. Increasing the spread of the particle set, however, also increases the number of particles required to effectively sample the search area. To make explicit and accelerate the process of seeking the global maxima we incorporate a Kernel Mean-Shift tracking step at each stage in the annealing run.

The Kernel Mean Shift tracker [1] maintains a single estimate of target position, hill climbing from the previous location estimate toward a local minimum in the Bhattacharya distance between normalised, kernel weighted colour histograms representing the object model and local image data. Assuming a 3D colour histogram the Bhattacharya distance between model and candidate is:

$$bhata() = \sqrt{1 - \sum_i^L \sum_j^L \sum_k^L \sqrt{p(i,j,k) \times d(i,j,k)}} \quad (5)$$

where p and d are the object and the candidate models respectively. Kernel Mean Shift provides efficient and effective tracking in its own right as long as the target object does not move further than its own diameter or leave the search area between frames, and a number of variations on the theme have been described.

The iterative Mean Shift operation is as follows [1]:

$$x = \frac{\sum_i^L \sum_j^L \sum_k^L \sqrt{\frac{p(i,j,k)}{d(i,j,k)}} \times i}{wt}$$

(6)

$$y = \frac{\sum_i^L \sum_j^L \sum_k^L \sqrt{\frac{p(i,j,k)}{d(i,j,k)}} \times j}{wt}$$

where

$$wt = \sum_i^L \sum_j^L \sum_k^L \sqrt{\frac{p(i,j,k)}{d(i,j,k)}} \quad (7)$$

and x and y are the coordinates of the next estimate of the position of the centre of the object. In the current implementation the object and candidate are $10 \times 10 \times 10$ bin histograms recording RGB colour values. The histogram is normalised to sum to 1. We use a linear kernel having maximum weight at the centre of the circular target area and zero weight at boundaries and beyond.

In the proposed algorithm, which we term the Kernel Annealed Mean-Shift (KAMS) tracker, Kernel Mean Shift is used within each annealing run of an annealed particle filter to cluster particles back together after they have been spread apart by the addition of Gaussian noise. The KAMS algorithm is summarised in Figure 1 and its operation illustrated in Figure 2.

Kernel Annealed Mean Shift Tracker:

1. Acquire frame at time t_k , having a set $S_{k,M}$ of N unweighted particles from the previous time step,
2. Set weighting function index $m = M$
3. While ($m > 0$)
 - a. Assign each particle a weight $\pi_{k,m}^{(i)}$
 - b. Select N particles with replacement and add Gaussian noise:
 $S_{k,m-1}^{(i)} = S_{k,m}^{(i)} + B_m$
 - c. Apply Mean shift to each particle until the Bhattacharya distance between the model and image measured by the weighting function $W_{m-1}(Z_k, S_{k,m-1}^{(i)})$ becomes stable or minimum.
 $bhata = \sqrt{1 - \sum_i^M \sum_j^M \sum_k^M \sqrt{p(i,j,k) \times d(i,j,k)}}$
 - d. $m = m-1$
4. Go to 1.

Figure 1. The Kernel Annealed Mean-Shift (KAMS) tracking algorithm

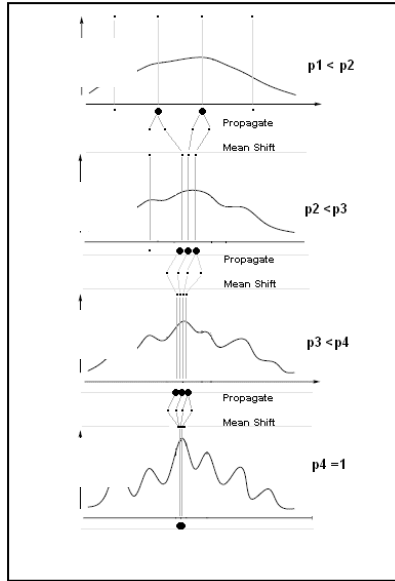


Figure 2. Visual representation of the KAMS filter in operation with four annealing stages.

The original annealed particle filter used sum of squared difference as its base weighting function. The Kernel Mean-Shift algorithm relied upon Bhattacharya distance. We employ Bhattacharya distance throughout. Kernel mean shift is run until Bhattacharya distance either falls below a small threshold or becomes stable. Experience has shown that this usually occurs within five iterations, so a limit on the number of iterations applied can reasonably be used, if necessary, to reduce computation.

3. Experimental Evaluation

3.1. Algorithms

The proposed KAMS tracker has been experimentally compared with the hybrid tracking algorithms proposed by Maggio and Caravello [6] and Naeem et .al [7].

In Maggio and Caravello's hybrid tracker (henceforth simply "Hybrid") Condensation provides a harness into which the Kernel Mean Shift tracker outlined in section 2 is slotted. At each time step particles are evaluated by computing the Bhattacharya distance between the object and their candidate model. Particles are then selected with probability proportional to their measurement value and projected into the next image by a constant velocity motion model. A kernel Mean Shift tracker is initialised at each particle location and run until its associated Bhattacharya distance becomes (approximately) zero or constant. A limit on the number of mean-Shift iterations may again be imposed to reduce computation without significant degradation in performance.

Naeem et. al.'s [7] Structured Octal Kernel algorithm (henceforth "SOK") is a kernel Mean Shift tracker augmented by a backup strategy triggered when confidence in the current location estimate is low. Confidence at time t is given by

$$C_t = (1.0 - bhata(t)) \quad (8)$$

where $bhata(t)$ is the Bhattacharya distance between object model and image data at time t . A user-defined threshold, T , is applied to C at each time step. If C_t is below threshold a set of eight independent kernel Mean Shift trackers are spawned, each with the same object model as the original but at locations designed to cover a search area around the current position estimate (Figure 3). When these additional trackers have also each converged, nine estimates of target location are available, each with an associated confidence level. The estimate with the highest confidence is selected and the process continues. In the original formulation the object model was a two-dimensional histogram of red/blue and green/blue. The implementation employed here uses a $10 \times 10 \times 10$ RGB histogram.

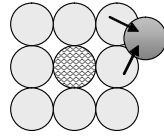


Figure 3. The SOK algorithm in operation. A hatched circle shows the primary Kernel Mean-Shift tracker, light circles the secondary “particles”, a dark circle the target.

3.2. Robustness

Quantitative, comparative analysis of the robustness of the proposed KAMS algorithm is achieved using McNemar’s statistic [8]. McNemar’s statistic is a form of chi-square test for matched paired data. Consider the following 2×2 table of results for two algorithms:

	Algorithm A Failed	Algorithm A Succeeded
Algorithm B Failed	N_{ff}	N_{sf}
Algorithm B Succeeded	N_{fs}	N_{ss}

Table 1. Terminology used in McNemar’s test – N_{xy} gives the number of times algorithm A produced result x and algorithm B produced result y , f and s denote failure and success respectively.

Mc Nemar’s statistic is then:

$$x^2 = \frac{(|N_{sf} - N_{fs}| - 1)^2}{N_{sf} + N_{fs}} \quad (9)$$

The Z score (standard score) is obtained as:

$$z = \frac{(|N_{sf} - N_{fs}| - 1)}{\sqrt{N_{sf} + N_{fs}}} \quad (10)$$

If the two algorithms give similar results then Z will tend to zero. As their results diverge, Z increases. Confidence limits can be associated with the Z value [8].

To apply McNemar, a definition of success and failure is required. Focussing on robustness, and recognizing that any tracker will fail at some point, we consider algorithm A to have succeeded and algorithm B to have failed if algorithm A maintains tracking for a greater proportion of a given image sequence, from the same starting parameters. In effect we define success to be tracking as long as the more successful of the two trackers.

McNemar's test was applied to a set of 30 assorted image sequences to provide quantitative comparison of the robustness of KAMS with Hybrid and SOK. With Z scores of 2.939 and 2.219 respectively, KAMS is significantly more robust than Hybrid with a confidence of 99.5%, and significantly more robust than SOK with a confidence of 98.5%.

Figure 4 shows selected frames from the three algorithms' tracking of one of the sequences used in the McNemars test, which shows a tiger sprinting through dense jungle. The animal's motion is smooth, but quite fast and there are frequent changes in head direction. The surrounding trees generate many partial occlusions and the darker stripes on the animal and the dark shadows caused by leaves in the background are similar, generating high levels of potentially confusing background clutter.

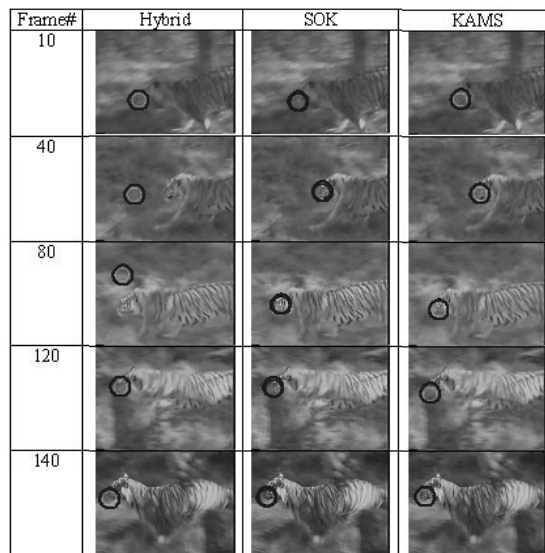


Figure 4. Hybrid, SOK and KAMS track a sprinting tiger

Hybrid fails around frame 30; the frequent changes in head velocity violate its motion model. SOK and KAMS both track robustly till the end of the video (frame 140). Neither employs a motion model, and their combined use of particles and mean-shift allow them to use a large enough search space to keep the tiger's head within bounds. At nine times the area of the target the search area used by SOK is very large indeed revealing the brute-force nature of its search. Though KAMS uses more particles than SOK, annealing keeps them focused around the tiger's head. All algorithms were manually initialised to the same point and (except SOK) used 200 particles.

Figure 5 shows the three algorithms tracking a table tennis ball being bounced on a bat. Hybrid (200 particles), fail around the 30th frame as its motion model is violated by the sharp changes in direction at the top of the ball's path. Though it hovers around after failure

and latches on to the again, this effect cannot be relied upon. The mean-shift tracker driving SOK copes well with the first change of direction at the top of the bounce, but loses confidence as the ball accelerates towards the bat. The structured search space is triggered, but, as it is fixed by the target size, cannot respond to the target's acceleration and loses track. KAMS maintains tracking during bounces and as the camera zooms out. Moreover, it achieves this using only 50 particles, and a much smaller search area than SOK.

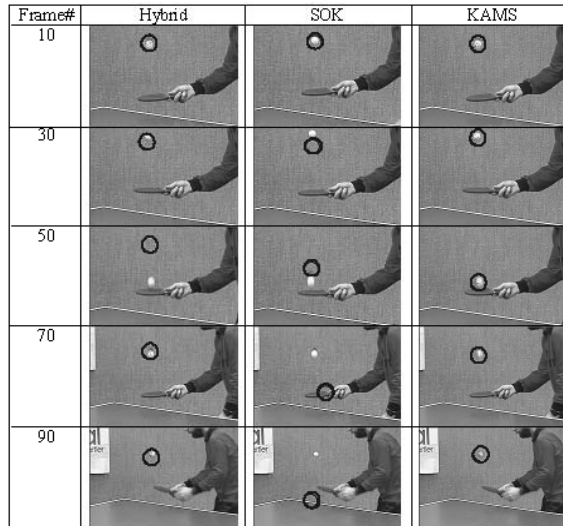


Figure 5. Hybrid, SOK and KAMS track a bouncing ball

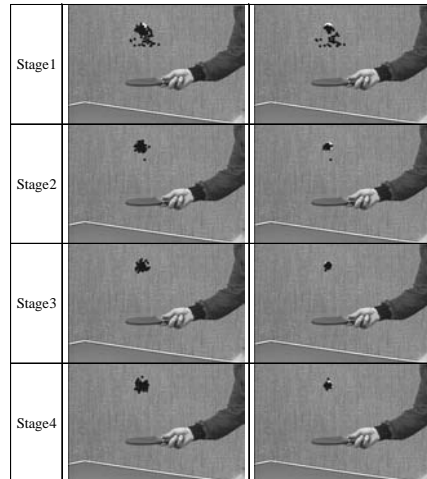


Figure 6. Alternate dispersal and clustering of particles during an annealing run in KAMS.

Figure 6 shows the set of particles generated at each stage of a single annealing run in KAMS. Each row shows the two particle sets created for a single value of M . The left image shows the particles after addition of Gaussian noise, the right after mean-shift. Note the alternating expansion and contraction of the particle set. Note also that mean-shift creates near-constant patterns of particles at the second annealing step. Space restrictions prevent a detailed examination of the effect of varying M , but experience to date suggests that KAMS will require fewer annealing levels than pure annealed particle filtering.

3.3. Accuracy

Artificial sequences showing a multicoloured circular target moving across a static background allow the trackers' positional estimates to be compared to ground truth in the presence of controlled amounts of measurement noise and clutter. Noise is simulated by perturbing the target's position in each frame with additive Gaussian noise. Clutter is added by randomly placing a user-defined number of similar circular objects on the otherwise white background (Figure 7). These distracting objects introduce local maxima into the evaluation function, while increased measurement noise raises the likelihood that a given tracker will come into contact with those maxima. All the artificial sequences used here consist of 140 (320x240 pixel) frames.

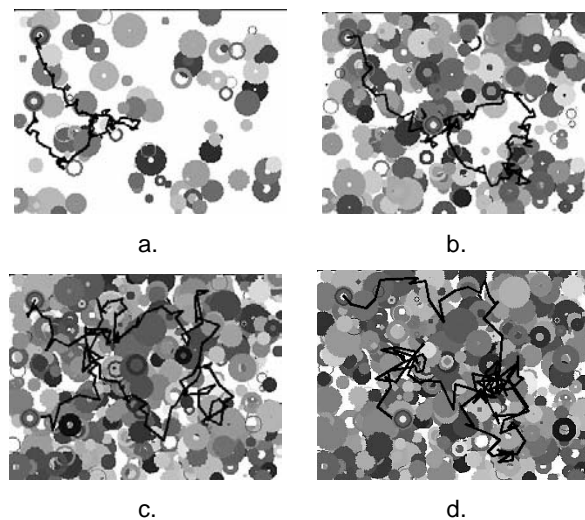


Figure 7. Sample images from the artificial test sequences, a. $\sigma = 4$ with 100 background objects, b. $\sigma = 8$ with 300 objects, c. $\sigma = 12$ with 500 objects, and d. $\sigma = 14$ with 600 objects. Black lines show sample paths, with the target displayed at either end.

Hybrid completed the sequence exemplified in figure 7a with a mean error of 6.32 pixels, but failed around frame 20 when noise and clutter increased. SOK completed figures 7a. and b. with mean errors of 5.66 and 9.60 pixels, but failed thereafter. Only KAMS managed to track through all 4 sequences, with mean errors of 6.94, 18.17, 14.72 and 17.30 pixels. The algorithms produced similar levels of accuracy (where comparable data is available), but KAMS is noticeably more robust to noise and clutter. Note also that Hybrid used 100 particles and KAMS only 40. KAMS manages particles more efficiently than Hybrid and so requires significantly fewer.

4. Conclusion

Hybrid tracking algorithms which combine particle filtering with the kernel mean-shift tracker have been shown to have performance advantages over their component parts. They also have computational advantages; by centring particles over local extrema the mean-shift stage reduces the number of particles required.

Although the efficiency gains are significant, we believe the key feature of hybrid trackers to be their exploitation of the natural tension between the process noise of the particle filter and the clustering performed by kernel mean-shift. We suggest that this tension provides opportunities to better manage the spread of particles across the search space, providing high performance with fewer particles.

We have proposed a novel hybrid tracker (KAMS) that combines kernel mean-shift [1] with the annealed particle filter [3]. The accuracies achieved by the various hybrid algorithms are comparable. The proposed algorithm, however, is significantly more robust than both previous hybrids and requires noticeably fewer particles than Maggio and Caravello's [6] algorithm.

Note also that KAMS does not include an explicit model of direction of movement, requiring only that the target stay within the search area defined by the process noise of the annealed particle filter. Mean shift allows that search area to be increased, enabling the algorithm to track successfully during sudden changes in velocity. KAMS may therefore be employed when a sufficiently simple and reliable motion model is hard to identify, and has been shown to perform well in situations (Figures 5 and 6) that have previously required multiple motion models [9].

5. References

- [1] D. Comaniciu, V. Ramesh, and P. Meer, Kernel-based object tracking, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(5), pp. 564–577, 2003.
- [2] M. Isard and A. Blake, CONDENSATION – conditional density propagation for visual tracking, *International Journal of Computer Vision*, 29(1) pp5-28, 1998.
- [3] J. Deutscher, A. Blake, and I. Reid, Articulated body motion capture by annealed particle filtering, *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2000.
- [4] J. Vermaak, A. Doucet and P. Perez, Maintaining multi-modality through mixture tracking, *Proc. ICCV*, pp 1110-1116, 2003.
- [5] C. Chang and R. Ansari. Kernel particle filter for visual tracking, *IEEE Signal Processing Letters.*, 12(3), pp. 242–245, 2005.
- [6] E. Maggio and A. Cavallaro, Hybrid particle filter and Mean Shift tracker with adaptive transition model, *Proc. Int. Conf. Acoustics, Speech, and Signal Processing 2005*.
- [7] A. Naeem, S. Mills, and T. Pridmore, Structured Combination of Particle Filter and Kernel Mean-Shift Tracking, *Proc. IVCNZ, Gt. Barrier Island*, 2006
- [8] A Clark and C. Clark, Performance Characterisation in Computer Vision – A Tutorial, <http://peipa.essex.ac.uk/benchmark/tutorials/essex/tutorial.pdf>.
- [9] M. Isard, and A. Blake, A mixed-state Condensation tracker with automatic model-switching, *Proc 6th Int. Conf. Computer Vision*, pp. 107-112, 1998.