View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by IntechOpen

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

15

Continuous Machine Learning in Computer Vision – Tracking with Adaptive Class Models

Rustam Stolkin
Stevens Institute of Technology
USA

1. Introduction

A fundamental (and popular) task in computer and robot vision is the tracking of an object which moves relative to the camera, essentially segmenting the object region of each successive frame. There are a great many published approaches, which are often variations, combinations or advances on well known techniques such as background subtraction, image differencing, predictive filtering and Bayesian estimation. Generally, these techniques rely on simple models of the tracked object and/or models of the background.

Many techniques in computer vision derive from ideas previously established in the pattern recognition community, where it is usual to learn models offline from historical training data sets. Hence these models, once learned, typically remain static during the online tracking process.

Such static models are ultimately of limited robustness in real world computer vision tracking scenarios where the appearance of both the background and the tracked object may change significantly and frequently due to camera motion (resulting in background change), object motion or deformation, introduction and removal of additional objects and clutter (e.g. passing traffic on a road) and changes in lighting and visibility conditions (either changes in ambient conditions or, for example, spotlights mounted on and moving with an underwater robot).

In contrast, this chapter will discuss a variety of tracking algorithms and techniques which are highly adaptable. These techniques have in common that they incorporate models which are continuously relearned from new input image frames while simultaneously performing tracking on those frames.

These techniques are powerful, in that they offer a way of successfully adapting to a changing environment. However, the price paid for adaptability can be a tendency towards certain kinds of instability. In simple terms, any system that continuously relearns (e.g. models of the tracked object and the background), has a risk of relearning incorrectly (e.g. relearning that background looks like object). Therefore, this chapter will also discuss various techniques for automatically detecting and correcting such errors as they occur, and survey techniques by which algorithms might continuously monitor their own performance.

It is also useful to consider continuous machine learning techniques in vision in terms of the rate of relearning. Firstly we will consider well established algorithms which incrementally re-learn models, very gradually, over many frames. Later we will look at very recent work,

Open Access Database www.i-techonline.com

Source: Scene Reconstruction, Pose Estimation and Tracking, Book edited by: Rustam Stolkin,
ISBN 978-3-902613-06-6, pp.530, I-Tech, Vienna, Austria, June 2007

in which models are entirely relearned at every frame or even several times during an iterative analysis of each frame. We will see that this re-learning rate often has implications for the trade off between the capacity of an algorithm to adapt and its inherent stability.

2. Adaptive background subtraction with a stationary camera

2.1 Relearning simple uni-modal background models

If the camera is fixed (e.g. in visual surveillance applications), and the tracked object is also moving, the simple but powerful technique of background subtraction can be employed in order to segment the image region representing the tracked object of interest. Typically, this involves thresholding the difference between the current image and a historical model of what the image looked like before any objects of interest were present. In its simplest form, this relies on the assumption that background pixel values remain constant. While this assumption can be effective for short term tracking in indoor environments with fixed lighting, it fails in longer term use in changing environments, especially for outdoor scenes which involve lighting and shadow changes, repetitive motion of clutter or slowly acting long-term changes to the scene. Thus it becomes desirable to enable the background model to gradually be re-learned.

A simple approach (Kanade et al., 1998, Collins et al., 1999) involves, essentially, modelling each background pixel intensity as a weighted moving average of recent pixel values. At the t^{th} video frame, the grey-scale intensity, I_{ij}, of each pixel, (i, j), is examined. If it is determined that this pixel represents background then the background model intensity, B_{ij}, for that pixel is updated as:

$$B_{ij}^{t+1} = \alpha B_{ij}^{t} + (1-\alpha) I_{ij}^{t} \tag{1}$$

otherwise the background model for that pixel is left unchanged. Classification of each pixel is determined by thresholding the difference between its intensity and that of the current background model, i.e. the pixel is classified as foreground if:

$$\left| I_{ij} - B_{ij}^{t} \right| > T_{ij}^{t} \tag{2}$$

Each pixel is assigned its own individual threshold, T_{ij}^{t}, which can itself be updated to take account of increases or decreases in the amount of temporal variation of background intensity. If a pixel is classified as background, then its threshold is updated as:

$$T_{ij}^{t+1} = \alpha T_{ij}^{t} + (1-\alpha)\left(\beta \times \left| I_{ij}^{t} - B_{ij}^{t} \right|\right) \tag{3}$$

i.e., the threshold for classifying foreground pixels is increased if the variation of background intensity from frame to frame increases and is decreased as this temporal variation decreases. Thus the threshold, T_{ij}^{t}, is analagous to β times the local temporal standard deviation of intensity. This process effectively moderates the fundamental tradeoff between false positives (erroneously classifying background pixels as foreground due to the threshold being too low) and false negatives (erroneously classifying foreground pixels as background because the threshold is set too high).

2.2 Relearning multi-modal background models

The above method is useful in that it continuously adapts to a (slowly) varying background scene. However, the simple uni-modal model cannot adequately account for the multi-modality that typically occurs in background pixels of real scenes, even when the camera is stationary. As an example, consider a fixed surveillance camera where a small part of the image views the branch of a tree. As the tree (or even the camera mounting) sways in the wind, a particular pixel colour might vary between blue (sky) and brown (tree branch). In such a situation we might wish for a bi-modal background model which can represent both of these common pixel values. A tri-modal model might further enable us to handle scenes in which background pixels typically represented tree branch, blue sky, or grey cloudy sky. Such multi-modal background variation occurs for myriad reasons, e.g. reflections from a rippling water surface in an outdoor scene or computer monitor flicker in an indoor office environment.

A continuosly relearnable model, which both addresses the multi-modality in background pixels and also adapts itself to temporal background changes, was first developed by Grimson and Stauffer (Grimson et al., 1998, Stauffer and Grimson, 1999, Grimson et al., 2000). Grimson models the recent history of each pixel colour (e.g. rgb value) over the previous t frames, $\{\overline{\mathbf{C}}_0,...,\overline{\mathbf{C}}_{t-1}\}$, as a mixture of K Gaussian distributions. The probability of observing the current pixel colour is:

$$P(\overline{\mathbf{C}}_t) = \sum_{k=1}^{K} \omega_{k,t-1} N(\overline{\mathbf{C}}_t, \overline{\boldsymbol{\mu}}_{k,t-1}, \boldsymbol{\Sigma}_{k,t-1}) \tag{4}$$

where $\omega_{k,t}$ is a weight (that portion of the data which is represented by this Gaussian) of the k^{th} of K Gaussians at time t, $\overline{\boldsymbol{\mu}}_{k,t}$ and $\boldsymbol{\Sigma}_{k,t}$ are the means and covariance matrices respectively and N denotes the Gaussian probability density function. At each frame, every new pixel value is checked against the K Gaussians and assigned to the best match. If none of them match (e.g. the pixel does not lie within 2 standard deviations of any Gaussian) then the least probable Gaussian is removed and replaced with a new Gaussian having the current value as its mean, a high initial variance and a small weight.

Now the weights of all K Gaussians are updated as:

$$\omega_{k,t} = \begin{cases} (1-\alpha)\omega_{k,t-1} + \alpha & \text{if the new pixel belongs to the } k^{\text{th}} \text{ Gaussian} \\ (1-\alpha)\omega_{k,t-1} & \text{otherwise} \end{cases} \tag{5}$$

After this update the weights are all re-normalized. α determines the rate at which the background model is relearned and has important consequences which we will discuss in more detail later. The weights, $\omega_{k,t}$, could be thought of as prior probabilities of each kind of background mode (e.g. the tree branch or the sky). Analagous with recursive Bayesian filtering, this prior has, in effect, been approximated as a weighted average of the previous posterior probabilities, with exponentially decaying emphasis on past values.

Grimson also seeks to adaptively relearn the means and variances of each Gaussian in the mixture. A simplifying (though usually untrue) approximation is to assume that red, blue and green components of each pixel are independent and share similar variances, i.e.:

$$\mathbf{\Sigma}_{ki,t} = \sigma_{k,t}^2 \mathbf{I} \tag{6}$$

Values of $\overline{\boldsymbol{\mu}}_{k,t}$ and $\sigma_{k,t}^2$ for those Gaussians which do not match the current pixel value remain unadjusted. Values of $\overline{\boldsymbol{\mu}}_{k,t}$ and $\sigma_{k,t}^2$ for the Gaussian to which the new pixel value does belong are updated as follows:

$$\overline{\boldsymbol{\mu}}_t = (1-\beta)\overline{\boldsymbol{\mu}}_{t-1} + \beta\overline{\mathbf{C}}_t \tag{7}$$

$$\sigma_t^2 = (1-\beta)\sigma_{t-1}^2 + \beta\left(\overline{\mathbf{C}}_t - \overline{\boldsymbol{\mu}}_t\right)^T\left(\overline{\mathbf{C}}_t - \overline{\boldsymbol{\mu}}_t\right) \tag{8}$$

The second learning rate, β, is simply the overall learning rate, α, weighted by the probability that the observed pixel value truly belongs to the Gaussian being modified, i.e.:

$$\beta = \alpha N\left(\overline{\mathbf{C}}_t, \overline{\boldsymbol{\mu}}_k, \mathbf{\Sigma}_k\right) \tag{9}$$

To determine the background model for each pixel individually, all *K* Gaussians for that pixel are ordered on the basis of ω_k / σ_k^2. This is a heuristic that assigns importance to modes which are both frequent and consistent (have a small variance). Once the Gaussians have been ordered in importance, the first *B* distributions are selected that account for a predefiend fraction, *F*, of observations, i.e.:

$$B = \arg\min_b\left(\sum_{k=1}^{b}\omega_k > F\right) \tag{10}$$

Now, any new pixel which is more than 2 standard deviations from the means of all of the *B* background distributions is classified as part of a foreground moving object.

Grimson and Stauffer's method provides a relatively sophisticated description of the background, which can be continuously relearned to enable powerful adaption capabilities for slowly changing scenes. A significant advantage of the method is that new characteristics of the background can be acquired without destroying the existing model. Statistically important colours will remain in the model until they become the K^{th} most probable mode and a new colour is observed. This enables for example, the background model to cope robustly with objects that move into the scene, temporarily stop, and then move on. Even if the stationary vehicle has temporarily been incorporated into the background model, it will quickly be removed again once it recomences motion.

2.3 Relearning non-parametric background models

Elgammal et al., 1999, suggest an alternative model for relearning backgrounds with a stationary camera. Grimson and Stauffer's mixture model approach builds a background model very slowly over a large number of image frames. This is unable to respond sufficiently sensitively to higher frequency background variations. To address this difficulty,

Elgammal et al. use a different kind of model that can be completely relearned over a much smaller, recent set of frames (good results are reported with 100 frames).
Given the colours of a pixel over the previous t frames, $\{\overline{\mathbf{C}}_0,...,\overline{\mathbf{C}}_{t-1}\}$, the probability density that this pixel will have rgb colour $\overline{\mathbf{C}}_t = (C_1^t, C_2^t, C_3^t)^T$ in the current frame can be non-parametrically estimated using a kernel estimator, K, as:

$$p(\overline{\mathbf{C}}_t) = \frac{1}{t}\sum_{i=1}^{t} K(\overline{\mathbf{C}}_t - \overline{\mathbf{C}}_i) \tag{11}$$

The kernel estimator function, K, is typically chosen to be a Normal function, $N(0,\boldsymbol{\Sigma})$, giving the density in terms of the multivariate Normal distribution:

$$p(\overline{\mathbf{C}}_t) = \frac{1}{t}\sum_{i=1}^{t} \frac{1}{(2\pi)^{\frac{3}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\overline{\mathbf{C}}_t-\overline{\mathbf{C}}_i)^T \boldsymbol{\Sigma}^{-1}(\overline{\mathbf{C}}_t-\overline{\mathbf{C}}_i)} \tag{12}$$

As with grimson's work, a simplifying approximation is to assume independence between the r,g and b colour values of the pixel so that:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix} \tag{13}$$

conveniently reducing the density estimation to:

$$p(\overline{\mathbf{C}}_t) = \frac{1}{t}\sum_{i=1}^{t}\prod_{j=1}^{3} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{1}{2}\frac{(C_j^t - C_j^i)^2}{\sigma_j^2}\right\} \tag{14}$$

This pixel is now labelled as foreground if $p(\overline{\mathbf{C}}_t) < T$, where T is a global threshold for the whole image. To estimate the variances for each colour, σ_j^2, the median, m_j, is computed of the deviations between each successive pair of values in the sample, e.g. the median, m_j, of $|C_j^n - C_j^{n-1}|$ for each consecutive pair of the previours 100 frames. Now each standard deviation is estimated as:

$$\sigma_j = \frac{m}{0.68\sqrt{2}} \tag{15}$$

2.4 Relearning rate

The background subtraction methods described so far are unable to detect objects which move slower than a critical "re-learning speed" since the object itself would simply become re-learned as background. Therefore there is a fundamental trade off in the choice of learning rate. It must be rapid enough to cope with the fastest anticipated background

change but slow enough to enable the most slowly moving objects to be detected. Thus two kinds of error must be considered when choosing an appropriate learning rate. Failing to relearn rapidly enough causes rapidly changing background pixels to be falsely detected as object (false positive). Failure to relearn slowly enough causes slow moving objects to pass undetected (false negative). In the extreme case, without the use of additional techniques, tracking based simply on background subtraction will eventually lose the tracked object if it stops moving since it will become incorporated into the relearned background model.

As an example of the complexity of these tradeoffs, Elgammal's model is able to adapt rapidly to background changes which would cause false positive detections with Grimson and Stauffer's method. However, Grimson and Stauffer's method is able to learn new background features without destroying its existing model. In contrast Elgammal's model is unable to remember background data from longer ago than 100 frames (or whatever length of frame history is chosen). Elgammal goes a certain way to overcoming this trade off by incorporating a procedure for combining both a short-term and long-term background model which are each updated over different timescales.

Since all of the methods so far described, including the faster responding non-parametric method, involve relatively slow relearning over many frames, they are all unable to cope with the rapidly changing backgrounds that result from a moving camera. The following sections describe recent work, in which rapid changes due to camera motion can be handled by a very different approach which enables the background to be completely relearned with every new frame.

3. The ABCshift algorithm – adapting backgrounds with a moving camera

3.1 Bayesian mean shift tracking with static colour models

The CAMSHIFT tracker (Bradski, 1998a, 1998b) is a colour based tracking algorithm which is popular for its elegant simplicity and speed. The qualities of speed and simplicity would suggest useful applications to mobile robot vision or wide area surveillance tasks which necessitate moving cameras. Unfortunately, CAMSHIFT was originally designed by Bradski for face tracking at close range from a stationary camera in relatively simple indoor environments. It often fails if the camera moves, because it relies on static models of both the background and the tracked object.

For each frame of an image sequence, the CAMSHIFT algorithm looks at pixels which lie within a subset of the image defined by a search window (green box in figures 1-5). Each pixel in this window is assigned a probability that it belongs to the tracked object, creating a 2D distribution of object location over a local area of the image. The centroid of this distribution can be regarded as the probabilistic expectation of the true object position, and thus provides an improved object position estimate. The search window is now repositioned at this centroid and the process is iterated until convergence. Since this iterative shift towards the mean (expectation) position is an example of the mean shift procedure (Comaniciu, 2002, 2003), Bradski's algorithm is known as the "Continuously Adaptive Mean Shift" or CAMSHIFT tracker. However, Bradski's use of the term "adaptive" is not the same as that of this chapter and does not imply any continuous machine learning. CAMSHIFT is only "adaptive" in the sense that the tracked object size is re-estimated at each frame to indicate whether the object is moving towards or away from the camera.

The size of the tracked object region (in pixels) is estimated by summing the probabilities of all the pixels within the search window. The object region can now be indicated by marking out a simple area of this size around the object centroid (the red box in figures 1-5). The search window is now resized so that its area is always in a fixed ratio to this estimated object area.

The tracked object is modeled as a class conditional colour distribution, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$. Depending on the application, 1D Hue, 3D normalised RGB, 2D normalised RG, UV or ab histograms may all be appropriate choices of colour model, the important point being that these are all distributions which return a probability for any pixel colour, given that the pixel represents the tracked object. These object distributions can be learned offline from training images, or during initialisation, e.g. from an area which has been user designated as object in the first image of the sequence.

The object location probabilities can now be computed for each pixel using Bayes' law as:

$$\mathrm{P}(O \mid \overline{\mathbf{C}}) = \frac{\mathrm{P}(\overline{\mathbf{C}} \mid O)\mathrm{P}(O)}{\mathrm{P}(\overline{\mathbf{C}})} \tag{16}$$

where $\mathrm{P}(O \mid \overline{\mathbf{C}})$ denotes the probability that the pixel represents the tracked object given its colour, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$ is the colour model learned for the tracked object and $\mathrm{P}(O)$ and $\mathrm{P}(\overline{\mathbf{C}})$ are the prior probabilities that the pixel represents object and posesses the colour, $\overline{\mathbf{C}}$, respectively.

The denominator of equation (16) can be expanded as:

$$\mathrm{P}(\overline{\mathbf{C}}) = \mathrm{P}(\overline{\mathbf{C}} \mid O)\mathrm{P}(O) + \mathrm{P}(\overline{\mathbf{C}} \mid B)\mathrm{P}(B) \tag{17}$$

where $\mathrm{P}(B)$ denotes the prior probability that the pixel represents background.

Bradski recommends values of 0.5 for both $\mathrm{P}(O)$ and $\mathrm{P}(B)$. However, this choice is difficult to justify if one takes these terms to denote the expected fractions of the total search window area containing object and background pixels respectively. It seems preferable to assign values to object priors in proportion to their expected image areas. If the search window area is always resized to be r times bigger than the estimated tracked object area, then $\mathrm{P}(O)$ is assigned the value $1/r$ and $\mathrm{P}(B)$ is assigned the value $(r-1)/r$.

The colour histograms, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$ and $\mathrm{P}(\overline{\mathbf{C}} \mid B)$, are the class conditional object and background models respectively. As for the object model, Bradski also suggests learning the background model offline, presumably building a static $\mathrm{P}(\overline{\mathbf{C}} \mid B)$ histogram from an initial image. While it is often reasonable to maintain a static distribution for the tracked object (since objects are not expected to change colour), a static background model is unrealistic when the camera moves. The CAMSHIFT algorithm can rapidly fail when the background scenery changes since colours may exist in the new scene which did not exist in the original distribution, such that the expressions in Bayes law will no longer hold true and calculated probabilities no longer add up to unity.

Particular problems arise with CAMSHIFT if the tracked object moves across a region of background with which it shares a significant colour. Now a large region of background may easily become mistaken for the object, figure 1.

3.2 Incorporating an adaptive background model

Recent work (Stolkin et al. 2006) addresses these problems by using a background model which can be continuously relearned. An interesting aspect of the work is that, in contrast to Grimson and Stauffer's mixture model representation (section 2.2), this model can be relearned without the need to decisively classify pixels as being object or background. Due to the continuously relearnable background model, Stolkin et al. have named this tracker the ABCshift (Adaptive Background CAMSHIFT) algorithm.

Rather than using an explicit $P(\overline{\mathbf{C}} \mid B)$ histogram, Stolkin et al. build a $P(\overline{\mathbf{C}})$ histogram which is recomputed every time the search window is moved, based on all of the pixels which lie within the current search window. $P(\overline{\mathbf{C}})$ values, looked up in this continuously relearned histogram, can now be substituted as the denominator for the Bayes' law expression of equation 16. Since the object distribution, $P(\overline{\mathbf{C}} \mid O)$, remains static throughout the tracking, this process becomes equivalent to implicitly relearning the background distribution, $P(\overline{\mathbf{C}} \mid B)$, because $P(\overline{\mathbf{C}})$ is composed of a weighted combination of both these distributions (see equation 17). Relearning the whole of $P(\overline{\mathbf{C}})$, rather than explicitly relearning $P(\overline{\mathbf{C}} \mid B)$, avoids the need to make hard decisions about the class of any particular pixel and helps ensure that probabilities add up to unity,e.g. if there are small errors in the static object model, $P(\overline{\mathbf{C}} \mid O)$.

Adaptively relearning the background distribution helps prevent tracking failure when the background scene changes, particularly useful when tracking from a moving camera (figures 1-4). Additionally, it enables objects to be tracked, even when they move across regions of background which are the same colour as a significant portion of the object, (figure 1-4). This is because, once $P(\overline{\mathbf{C}})$ has been relearned, the denominator of Bayes' law (equation 16) ensures that the importance of this colour will be diminished. In other words, the tracker will adaptively learn to ignore object colours which are similar to the background and instead tend to focus on those colours of the object which are most dissimilar to whatever background is currently in view.

It is interesting to note that the continual relearning of the $P(\overline{\mathbf{C}})$ histogram need not substantially increase computational expense. Once the histogram has been learned for the first image it is only necessary to remove from the histogram those pixels which have left the search window area, and add in those pixels which have newly been encompassed by the search window as it shifts with each iteration. Provided the object motion is reasonably slow relative to the camera frame rate, the search window motion will be small, so that at each iteration only a few lines of pixels need be removed from and added to the $P(\overline{\mathbf{C}})$ histogram.

If the $P(\overline{\mathbf{C}})$ histogram is relearned only once every frame, the speed should be similar to that of CAMSHIFT. However, if the histogram is relearned at every iteration, some additional computational expense is incurred, since to properly exploit the new information it is necessary to recompute the $P(O \mid \overline{\mathbf{C}})$ values for every pixel, including those already analysed in previous iterations. In contrast, with the CAMSHIFT algorithm, $P(O \mid \overline{\mathbf{C}})$ values only ever need to be computed once for any pixel. Theoretically, updating at each iteration should

produce more reliable tracking, although good tracking results are observed with both options.

Figure 1. A simple blue and red chequered object, moving from a region of white background into a region of red background. CAMSHIFT fails as soon as the object moves against a background with which it shares a common colour. Frames 350, 360, 380, 400, and 450 shown. Green and red squares indicate the search window and estimated object size respectively. This movie, RedWhite1CAMSHIFT.avi, can be viewed at the project website (see references).

Figure 2. ABCshift tracks successfully. Frames 350, 360, 380, 400, and 450 shown. Green and red squares indicate the search window and estimated object size respectively. This movie, RedWhite1ABCshift.avi, can be viewed at the project website (see references).

Figure 3. Person tracking with CAMSHIFT from a moving camera in a cluttered, outdoors environment. Frames 1, 176, 735, 1631 and 1862 shown. Since the tracked person wears a red shirt, CAMSHIFT tends to fixate on red regions of background, including brick walls and doors, and repeatedly loses the tracked person. Green and red squares indicate the search window and estimated object size respectively. This movie, PeopleTracking1CAMSHIFT.avi can be viewed at the project website (see references).

Figure 4. ABCshift successfully tracks throughout the sequence and is not distracted by red regions of background, despite being initialised in image 1 which contains no red background. Frames 1, 176, 735, 1631, and 1862 shown. Green and red squares indicate the search window and estimated object size respectively. This movie, PeopleTracking1ABCshift.avi, can be viewed at the project website (see references).

In practice, ABCshift may often run significantly faster than CAMSHIFT. Firstly, the poor background model can cause CAMSHIFT to need more iterations to converge. Secondly, the

less accurate tracking of CAMSHIFT causes it to automatically grow a larger search window area, so that far greater numbers of pixels must be handled in each calculation.

3.3 Summary of the ABCshift tracker

The key difference between ABCshift and the conventional CAMSHIFT tracker is that CAMSHIFT uses a simple, static background model that is typically initialized from the first frame and then remains constant throughout the duration of the tracking. In contrast, ABCshift is able to completely relearn the background at every frame or even many times per frame, with little additional computational cost.

The ABCshift algorithm is summarized as:

1. Identify an object region in the first image and train the object model, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$.
2. Center the search window on the estimated object centroid and resize it to have an area r times greater than the estimated object size
3. Learn the colour distribution, $\mathrm{P}(\overline{\mathbf{C}})$, by building a histogram of the colours of all pixels within the search window.
4. Use Bayes' law (equation 16) to assign object probabilities, $\mathrm{P}(O \mid \overline{\mathbf{C}})$, to every pixel in the search window, creating a 2D distribution of object location.
5. Estimate the new object position as the centroid of this distribution and estimate the new object size (in pixels) as the sum of all pixel probabilities within the search window.
6. Repeat steps 2-5 until the object position estimate converges.
7. Return to step 2 for the next image frame.

4. Algorithms that detect and correct their own errors

4.1 Automatic online performance evaluation

Förstner, 1996, suggests that:

1. Vision systems should contain tools for self diagnosis and be able to estimate their own performance.
2. Vision systems should know their own limitations, detect their own cases of failure and be able to report failures and possible causes.
3. To enable such self diagnosis, quality measures need to be determined and specified for both algorithm inputs and outputs.

There seem to be two fundamental mechanisms by which a vision algorithm might determine when it is (or is likely to be) failing or performing sub-optimally. These can broadly be divided into techniques that examine the algorithm's inputs and those that examine its outputs. Firstly, it might be possible to test the kinds of tracking conditions under which a particular algorithm tends to fail. By comparing these with the current input data to the algorithm during tracking, it might be possible to infer when imminent failure or poor performance is likely. Secondly, it may be possible to apply quality measures to output features of the algorithm such as characteristics of an estimated trajectory or a learned representation of the tracked object. The first strategy is difficult since it would require an extensive survey of the performance of an algorithm on a large number of image sequences, under many conditions, as well as some way of characterizing and comparing the conditions, i.e. some metrics which summarise the nature of any input video sequence. This

approach is theoretical and, to the best of the author's knowledge has not so far been attempted. Therefore, this section briefly examines some simple techniques that attempt to implement the second of these strategies.

There are two reasons for including this material in the chapter. The concept of algorithms which continuously monitor their own performance and recognise and correct their own errors, seems intuitively to be closely related to the principals of continuous machine learning and autonomous adaptability which are the subject of this discussion. Additionally, these techniques are particularly useful to help correct certain kinds of instability, which occasionally result from continuous model relearning. Simplistically, if an algorithm is allowed to continuously relearn without supervision, there is always a danger that it will learn incorrectly (e.g. learning that background looks like object). Once this process begins and is left uncorrected, it can sometimes escallate, creating an unstable feedback situation which results in failure. This is a previously underexplored area of research. The intention of this section is to highlight some examples and suggest a few possible research directions in the hopes of stimulating further interest within the vision community.

4.2 Bhattacharyya resizing

The ABCshift algorithm is powerful, in that it can cope with rapidly changing backgrounds due to camera motion, by completely relearning a background model at every frame. However, this continual relearning itself can introduce a special mode of instability which occasionally causes problems. If the search window should shrink (due to the object region being temporarily underestimated in size) to such an extent that the boundaries of the search window approach the boundaries of the true object region, then the background model will be retrained predominantly using object pixels. This in turn will lead to many object pixels being assigned a high probability of belonging to the background and even more object pixels beome incoporated into the background model. Thus the estimated object region shrinks in size with a corresponding shrinking of the search window. This results in an unstable feedback cycle with the estimated object region and search window gradually (and unrecoverably) collapsing.

Stolkin et al., 2007, solve this problem by noting that, as the search window shrinks and approaches the size of the object region, the learned search window distribution, $\mathrm{P}(\overline{\mathbf{C}})$, must become increasingly similar to the static distribution known for the tracked object, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$. If this increasing similarity can be detected, then both the object region and search window can be easily resized, see figure 5, the correct enlargement factor being r, the desired ratio of search window size to object region size.

Several statistical measures exist for comparing the similarity of two histograms. Stolkin et al. utilise a Bhattacharyya metric (Bhattacharyya, 1943) sometimes referred to as Jeffreys-Matsusita distance (Jeffreys, 1946) which for two histograms, $p = \{p_i\}_{i \in \{1,2,\ldots K\}}$ and $q = \{q_i\}_{i \in \{1,2,\ldots K\}}$ is defined as:

$$d(p,q) = \sqrt{\sum_{i=1}^{K} \left(\sqrt{p_i} - \sqrt{q_i}\right)^2} \tag{18}$$

$0 \le d \le \sqrt{2}$. Note that this metric can easily be shown to be the same, modulo a factor of $\sqrt{2}$ as that referred to elsewhere in the literature (Comaniciu, 2002, 2003, Perez, 2002, Numiaro, 2002).

At each iteration of the ABCshift algorithm, Stolkin et al., 2007, evaluate the Bhattacharyya metric between the static object distribution, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$, and the continuously relearned search window distribution, $\mathrm{P}(\overline{\mathbf{C}})$ (which implicitly encodes the background distirbution, $\mathrm{P}(\overline{\mathbf{C}} \mid B)$. If the Bhattacharyya metric approaches zero, it is inferred that the search window is approaching the true object region size while the estimated object region is collapsing. Both windows are therefore resized by the factor r . In practice it seems useful to resize when the Bhattacharyya metric drops below a preset threshold. Useful threshold values typically lie between 0.2 and 0.7.

Note that, because of the special way that ABCshift implicitly relearns the background by relearning the $\mathrm{P}(\overline{\mathbf{C}})$ histogram, the Bhattacharyya metric is used to compare this histogram with the object model, $\mathrm{P}(\overline{\mathbf{C}} \mid O)$. In other kinds of algorithm, where the literal background distribution itself is available, it would be equally advantageous to measure the Bhattacharyya metric betwen $\mathrm{P}(\overline{\mathbf{C}} \mid O)$ and $\mathrm{P}(\overline{\mathbf{C}} \mid B)$.

This is an unusual application of the Bhattacharyya metric. It has previously become common in the vision literature (Comaniciu, 2002, 2003, Perez, 2002, Numiaro, 2002) to use this metric to evaluate the similarity between a candidate image region and an object distribution for tracking (i.e. comparing potential object with known object). In contrast, Stolkin et al., 2007, use the metric to compare an object distribution with a background distribution, inferring an error if the two begin to converge.

Figure 5. Bhattacharyya resizing. A simple red and blue checkered object is tracked across red, white and blue background regions by the ABCshift tracker, augmented with Bhattacharyya resizing. Frames 180, 200, 205, 206 shown. Due to rapid, jerky motion from frames 180 to 205, the search window has shrunk until it falls within the object region, risking relearning that background looks like object. ABCshift has detected this instability using the Bhattacharyya metric, and automatically corrects the estimated object region and search window size in frame 206. Green and red squares indicate the search window and estimated object size respectively. This movie, PeopleTracking1ABCshift.avi, can be viewed at the project website (see references).

4.3 Other kinds of online auto-performance evaluation

Other related work includes Correia and Pereira, 2002, 2003, and Erdem et al., 2004. This work is not explicitly concerned with the concept of algorithms that constantly monitor their own performance during tracking. However the techniques are applicable, since the authors

are broadly interested in performance evaluation without the need for ground truth data (which can be very difficult to generate, see Stolkin, 2006). Erdem divides these performance metrics, which could be used to auto-evaluate tracking performance online (without any external ground-truth data), into two classes as intra-object homogeneity and inter-object disparity, i.e. the tracked object should be consistent with itself but different from the background or other objects.

Intra-object homogeneity metrics might examine shape regularity, spatial uniformity, temporal stability and motion uniformity. The Bhattacharyya resizing technique described above is similar to inter-object disparity metrics, which evaluate colour or motion contrast between pixels, labelled as lying inside and outside the tracked object.

For tracking schemes which output a detailed segmentation of the tracked object, Erdem et al. suggest evaluating spatial colour contrast along object boundaries. Pairs of pixels are selected which lie slightly inside and outside the boundary of the estimated segmented object region. Then colour differences are evaluated along the object boundary. If the tracking algorithm enables a colour histogram of the tracked object to be re-calculated at each frame (e.g. by defining a segmented object region or by relearning a colour model), then this histogram can be compared with a smoothed or average histogram from several previous frames, to measure temporal consistency. It is also possible to evaluate the differences in motion vectors of points estimated to lie inside and outside the tracked object.

In the author's opinion, the use of such techniques, even in very simple ways, to enable tracking algorithms to detect their own errors or modify their parameters in response to deteriorating performance has so far received very little attention, and this would seem to be a useful and open area of ongoing research.

5. Continuously adaptive models of the tracked object

So far, this chapter has provided an overview of various examples of continuous machine learning in the context of background models which adapt with time. Adaptive tracking research predominantly focuses on dynamic relearning of background models, rather than foreground models, because it is often reasonable to assume that the appearance (e.g. colour distribution or texture) of a tracked object remains relatively constant during the tracking process. This section will examine the possibilities for creating algorithms with the additional capabilities of adapting to changes in the tracked object.

Might it be possible to create a simple colour based blob tracker which can track a chameleon? Or how about tracking a person who, while strolling down the street, pulls off a red jacket to reveal a yellow shirt underneath (or Clark Kent as he changes into Superman on the fly)? At present, these kinds of problems (or the similar problem of tracking "camouflaged" objects) tend to be approached with contour tracking, e.g. the ConDensation algorithm (Isard and Blake, 1996), but might fast and simple algorithms such as Mean Shift Tracking (Comaniciu et al., 2003) or ABCshift (Stolkin et al., 2007) be modified to handle these tasks by continuously relearning the appearance of the tracked object?

Let us consider the case of region based object tracking with representations of objects in the form of distributions of intensity, colour or other simple features. In order to update such distributions based on the intensities of pixels in each new frame, some decision must be made about whether or not each new pixel belongs to the tracked object. Note that ABCshift successfully adapts to a changing background without any explicit classification of background pixels, but this is enabled by the assumption of a static object model, combined

with some cunning manipulation of Bayes' law. It is relatively easy to update a background model given a static object model and presumably vice versa, but it is much less obvious how to mutually refine both models at the same time. A method is needed by which entirely new colours, which previously did not exist in the object at all, could still be acquired by the object model as they appear. It does not seem logically feasible to manage this without fitting some kind of boundary or contour around the tracked object region at each frame.

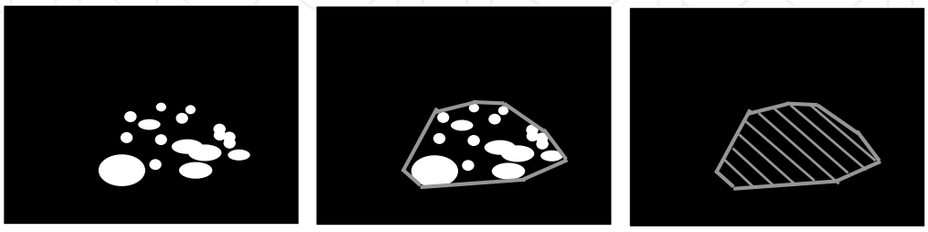


Figure 6. Possible mechanism for object model relearning. Firstly the old object model (e.g. colour distribution) is used to classify new image pixels as object. Next a boundary is fitted around these pixels, defining the object region. Finally all pixels within the object region are relearned as the new object model.

A simple theoretical example of such a scheme is illustrated in figure 6. If the object model is a class conditional intensity or colour distribution, then pixels in the current image, with high probabilities according to the existing model, can be identified as belonging to the object. Some object pixels are missed, since the object appearance has changed somewhat since the previous image and the current model is out of date. Hence it may be possible to estimate the object region by fitting a boundary around the detected object pixels. This boundary will include regions of colours which are missing from the current object model. The new colours can be incorporated by relearning the object model according to all pixels which lie in side the bounding contour. Depending on the application, this boundary might be a simple shape (e.g. an ellipse or a square), a flexible contour or snake or the projection of a known 3D model of a tracked rigid body.

An example of an algorithm which continuously relearns both object and background models, is the EM/E-MRF algorithm (Stolkin et al., 2000, 2007b, 2007c), which combines simple, Gaussian models of the object and background pixel intensities with a 3D CAD model of the rigid tracked object. The EM/E-MRF algorithm was an attempt to tackle images under conditions of extremely poor visibility, by combining observed image data with prior knowledge in various forms.

Given the recent trajectory of the camera relative to the observed object, a new relative camera pose is predicted for the current observed image. This pose is used to project a known 3D model of the object, yielding a prediction of the object region in the observed image. The projected/predicted object region can be used to roughly define those image pixels which represent the object. This then enables the creation of object and background distributions (1D Gaussians) from the observed image pixel intensities which lie in these regions.

The object and background distributions can now be used to segment the observed image (the EM/E-MRF algorithm probabilistically combines these distributions with predicted image data during segmentation, using the Extended-Markov Random Field procedure).

The 3D object model can now be best fitted to the segmented image to yield an improved camera pose estimate as well as a cleaner image segmentation. Now the camera pose can be recycled as a new input and the process is iterated. Camera pose, intensity distributions and image interpretation are mutually improved via an Expectation Maximisation-like iterative process. This iterative cycle is illustrated in figure 7.

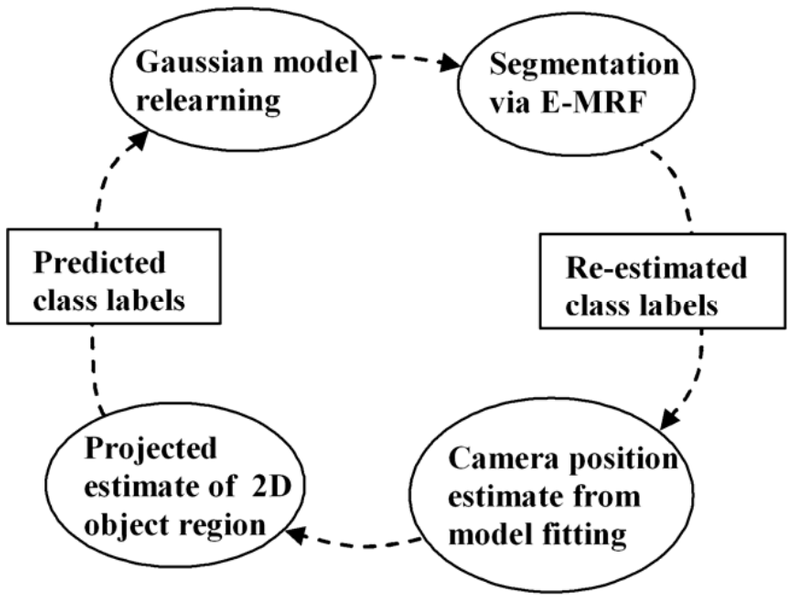


Figure 7. Iterative model relearning with the EM/E-MRF algorithm. Best fitting a projection of the tracked object provides a hard boundary to the estimated object region of the image. This enables object and background intensity distributions to be relearned accordingly.

The EM/E-MRF algorithm achieves some success at interpreting extraordinarily poor visibility images, due to the large amount of predicted information that it incorporates into the segmentation process. It is an interesting exercise in probabilistically fusing two images of the same scene, in this case an observed image and a predicted image. However, this algorithm uses object and background distributions which are overly simplistic for many scenes. At the same time, the Extended-Markov Random Field (E-MRF) optimisation, used in the segmentation process, is excessively computationally expensive.

Despite these drawbacks, the approach is included here as an interesting example of an algorithm which is able to simultaneously relearn both object and background distributions at every frame, see figure 8.

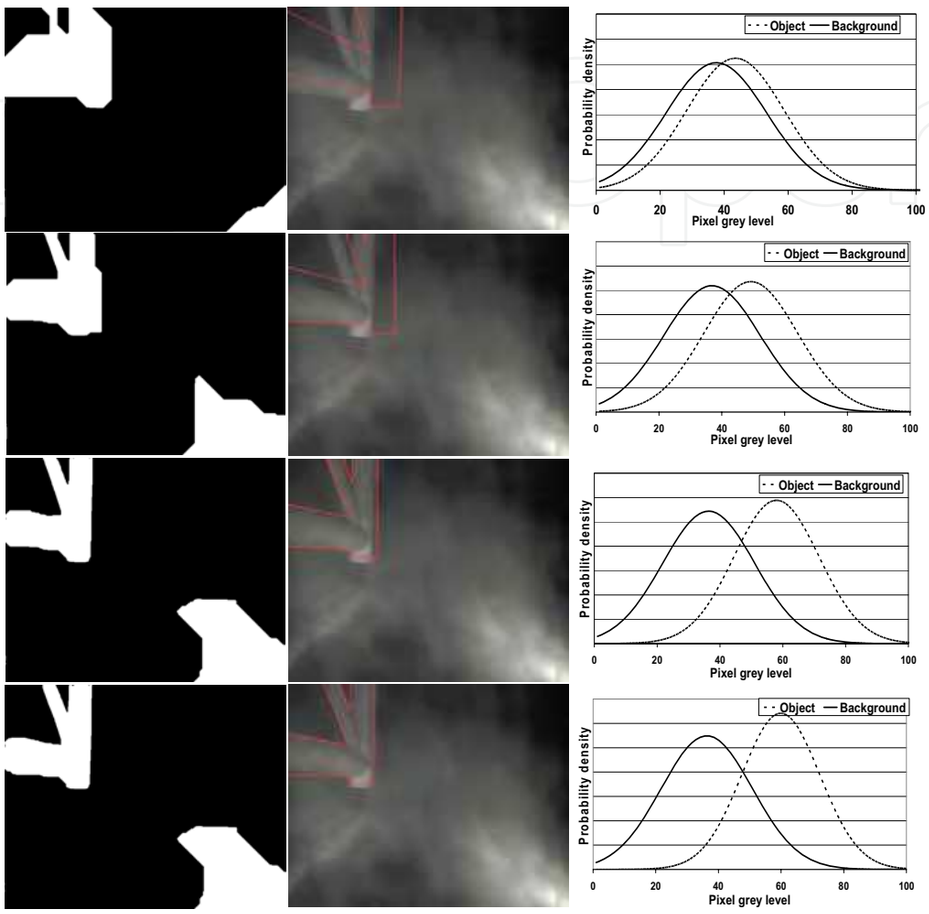


Figure 8. Example of EM/E-MRF tracking on a poor visibility image. Difficult, turbid conditions, encountered with underwater robotics, have been simulated in the lab with dry ice fog and focussed beam spot lights, mounted on and moving with the camera, leading to severe backscattering. Over four iterations, the EM/E-MRF algorithm homes in on an oil-rig like structure. Note how the algorithm mutually refines image interpretation, camera/object pose and the Gaussian object and background models. The two Gaussians separate as the algorithm progressively learns that object is relatively bright whereas background is relatively dark.

7. Conclusion

This chapter has explored a number of techniques in vision based tracking, with the common theme of continuous relearning of models of the background and the tracked

object. The majority of well known work in this area has focussed on relearnable background models for stationary cameras. These tend to be robust and stable, but adapt relatively slowly. More recent work has proposed models which can be completely relearned at every frame, enabling tracking with rapid camera motion and also robust tracking of objects which move across regions of background with which they share significant colours.

It is important to note that, although the ABCshift algorithm can relearn the background at every frame, it is only able to do so because it is initialised with a known, static model of the tracked object. In contrast, the adaptive models for stationary camera background subtraction adapt much more slowly, but are able to detect and track new objects without any prior information about the objects' appearance. This might suggest a hybrid scheme, whereby background subtraction techniques with stationary cameras are used to detect new objects and acquire their characteristics, and these characteristics are then passed to object model based techniques which can, for example, continue to follow objects of interest by servoing a pan, tilt, zoom camera.

In order to enable continuous relearning of object models in addition to background models, it seems necessary to define the object region in each frame by a contour or boundary. Thus region based tracking, effectively requires the fusion of contour tracking techniques in order to achieve full adaptability. It is interesting to note that these ideas naturally correspond with calls from elsewhere in the vision community (e.g. Blake, 2005) for hybrid trackers, incorporating both contours and regions, as a way forwards in robust tracking research.

8. References

ABCshift movies at http://www.math.stevens.edu/~ifloresc/ABCshift.htm, 2006.

A. Blake. (2005). Visual tracking: a short research roadmap. *Mathematical Models of Computer Vision: The Handbook,* Eds. O. Faugeras, Y. Chen and N. Paragios, Springer, in press.

Bhattacharayya, A. (1943) On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, Vol. 35, 1943, pages 99-110.

Bradski, G. (1998a). Computer video face tracking for use in a perceptual user interface, *Intel Technology Journal,* Q2, 1998.

Bradski, G. (1998b). Real time face and object tracking as a component of a perceptual user interface, *Proceedings of the 4th IEEE Workshop on Applications of Computer Vision,* pages 214-219, 1998.

Collins, R.; Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomotos, N., Hausegawa, O., Burt, P. & Wixson, L. (1999). A System for Video Surveillance and Monitoring, *Proceedings of the American Nuclear Society (ANS) Eighth International Topical Meeting on Robotic and Remote Systems,* April, 1999.

Comaniciu, D.; Meer, P. (2002) Mean shift: A robust approach toward feature space analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 24, No. 5, May 2002, pages 603-619.

Comaniciu, D.; Ramesh, V., Meer, P. (2003) Kernel-based object tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 25, 2003, pages 564-577.

Correia, P.; Pereira, P. (2002) Stand-alone objective segmentation quality evaluation. *EURASIP Journal of Applied Signal Processing*, No. 4, pages 390-402, 2002.

Correia, P.; Pereira, P. (2003) Objective evaluation of video segmentation quality. *IEEE Transactions on Image Processing*, Vol. 12, February, 2003, pages 186-200.

Elgammal, A.; Harwood, D., Davis, L. (1999) Non-Parametric Model for Background Subtraction. *Proceedings of the IEEE Frame Rate Workshop*, 1999.

Erdem, C.; Sankur, B., Tekalp, A. (2004) Performance Measures for Video Object Segmentation and Tracking. *IEEE Transactions on Image Processing*, Vo. 13, No. 7, July, 2004.

Förstner, W. (1996). Pros and cons against performance characterization of vision algorithms. *Proceedings of the European Conference on Computer Vision*, *Workshop on Performance Characteristics of Vision Algorithms*, 1996, pages 13-29.

Grimson, W.; Stauffer, C., Romano, R., Lee, L. (1998). Using adaptive tracking to classify and monitor activities in a site, *Computer Vision and Pattern Recognition*, June 1998.

Jeffreys, H. (1946). An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society (A)*, Vol. 186, pages 453-461.

Kanade, T.; Collins, R., Lipton, A., Burt, P. & Wixson, L. (1998). Advances in cooperative multi-sensor video surveillance, *Proceedings of the 1998 DARPA Image Understanding Workshop*, volume 1, 1998, pages 3-24.

Nummiaro, K.; Van Gool, L., Koller-Meier, E. (2002). Object tracking with an adaptive color-based particle filter. *Pattern Recognition : 24th DAGM Symposium*, *Proceedings*, Vol. 2449 of Lecture Notes in Computer Science, pages 353-360, 2002, Zurich, Switzerland.

Perez, P; Hue, C., Vermaak, J., Gangnet, M. (2002). Color-based probabilistic tracking. *Proceedings of the 7th European Conference on Computer Vision*, Vol. 2350, Part 1, Lecture Notes In Computer Science, 2002, pages 661-675.

Stauffer, C; Grimson, W. (1999). Adaptive background mixture models for real-time tracking. *Proceedings of Computer Vision and Pattern Recognition*, June 1999.

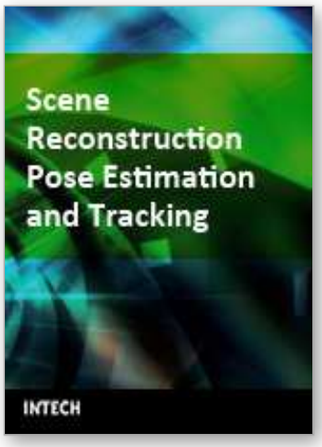
Stolkin, R.; Hodgetts, M., Greig, A. (2000). An EM/E-MRF strategy for underwater navigation, *Proceedings of the British Machine Vision Conference*, 2000.

Stolkin, R.; Greig, A., Gilby, J. (2006). A calibration system for measuring 3D ground truth for validation and error analysis of robot vision algorithms. *Journal of Measurement Science and Technology*. Institute of Physics, 2006.

Stolkin, R.; Florescu, I., Kamberov, G. (2007a). An adaptive background model for CAMSHIFT tracking with a moving camera. *Proceedings of the 6th International Conference on Advances in Pattern Recognition,* Kolkatta, January, 2007.

Stolkin, R.; Hodgetts, M., Greig, A., Gilby, J. (2007b). Extended Markov Random Fields for predictive image segmentation, *Proceedings of the 6th International Conference on Advances in Pattern Recognition,* Kolkatta, January 2007.

Stolkin, R., Greig, A., Hodgetts, M., Gilby, J. An EM / E-MRF algorithm for adaptive model based tracking in extremely poor visibility. *Journal of Image and Vision Computing*, Elsevier, 2007c.



Scene Reconstruction Pose Estimation and Tracking

Edited by Rustam Stolkin

ISBN 978-3-902613-06-6

Hard cover, 530 pages

Publisher I-Tech Education and Publishing

Published online 01, June, 2007

Published in print edition June, 2007

This book reports recent advances in the use of pattern recognition techniques for computer and robot vision. The sciences of pattern recognition and computational vision have been inextricably intertwined since their early days, some four decades ago with the emergence of fast digital computing. All computer vision techniques could be regarded as a form of pattern recognition, in the broadest sense of the term. Conversely, if one looks through the contents of a typical international pattern recognition conference proceedings, it appears that the large majority (perhaps 70-80%) of all pattern recognition papers are concerned with the analysis of images. In particular, these sciences overlap in areas of low level vision such as segmentation, edge detection and other kinds of feature extraction and region identification, which are the focus of this book.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Rustam Stolkin (2007). Continuous Machine Learning in Computer Vision - Tracking with Adaptive Class Models, Scene Reconstruction Pose Estimation and Tracking, Rustam Stolkin (Ed.), ISBN: 978-3-902613-06-6, InTech, Available from: http://www.intechopen.com/books/scene_reconstruction_pose_estimation_and_tracking/continuous_machine_learning_in_computer_vision_-_tracking_with_adaptive_class_models

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2007 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen