

Article

Robust Observation Detection for Single Object Tracking: Deterministic and Probabilistic Patch-Based Approaches

Mohd Asyraf Zulkifley ^{1,*}, David Rawlinson ² and Bill Moran ²

¹ Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia

² Department of Electrical and Electronic Engineering, The University of Melbourne, VIC 3010, Australia; E-Mails: davidjr@unimelb.edu.au (D.R.); wmoran@unimelb.edu.au (B.M.)

* Author to whom correspondence should be addressed; E-Mail: asyraf@eng.ukm.my; Tel.: +603-8921-6335.

Received: 18 September 2012; in revised form: 5 November 2012 / Accepted: 5 November 2012 / Published: 12 November 2012

Abstract: In video analytics, robust observation detection is very important as the content of the videos varies a lot, especially for tracking implementation. Contrary to the image processing field, the problems of blurring, moderate deformation, low illumination surroundings, illumination change and homogenous texture are normally encountered in video analytics. Patch-Based Observation Detection (PBOD) is developed to improve detection robustness to complex scenes by fusing both feature- and template-based recognition methods. While we believe that feature-based detectors are more distinctive, however, for finding the matching between the frames are best achieved by a collection of points as in template-based detectors. Two methods of PBOD—the deterministic and probabilistic approaches—have been tested to find the best mode of detection. Both algorithms start by building comparison vectors at each detected points of interest. The vectors are matched to build candidate patches based on their respective coordination. For the deterministic method, patch matching is done in 2-level test where threshold-based position and size smoothing are applied to the patch with the highest correlation value. For the second approach, patch matching is done probabilistically by modelling the histograms of the patches by Poisson distributions for both RGB and HSV colour models. Then, maximum likelihood is applied for position smoothing while a Bayesian approach is applied for size smoothing. The result showed that probabilistic PBOD outperforms the deterministic approach with average distance error of 10.03% compared with 21.03%. This algorithm

is best implemented as a complement to other simpler detection methods due to heavy processing requirement.

Keywords: tracking observation; Neyman–Pearson method; Poisson modelling; maximum correlation; histogram intersection; patch matching

1. Introduction

Obtaining the correct observation for track maintenance is a very challenging task. Tracking accuracy is highly dependent on accurate observation. Improving the accuracy of observation detection and association are two crucial factors in building good trackers, especially in people counting and behaviour analysis systems. Even for a global positioning system [1], observation data from several satellites are optimized in order to provide the best possible coordinate. During complex situations such as illumination changes, clutter and occlusion, robust observations are rarely obtained, which lead most trackers [2,3] to utilize prediction information only. Moreover, null observation and false association sometimes occur, which will hinder tracker performance. The challenge of detecting the tracked object throughout the video is more difficult compared with detecting an object in a database (image processing) due to the non-rigid nature of the object where its appearance varies with time. Even though there is a strong correlation between the current and previous frame in detecting the moving object, the template itself is not perfect, which makes comparison between frames very difficult. The dynamic change between frames is the reason behind the difficulty in establishing and retaining a good set of template. This paper is dedicated to robust observation detection, especially in various challenging environments and surroundings. The processing requirement of the algorithm is quite heavy, so we suggest the user implement a simpler detection algorithm as the primary method and only switch to our algorithm for challenging situations and surroundings. An example of a simple switching mechanism is if the main method fails to detect the observation, *i.e.*, there is no observation detected for that particular scan, then our algorithm is activated to obtain the measurement. We limit this paper to single object tracking only, which is a part of Zulkifley's PhD thesis [4].

Generally, there are two major approaches to obtain the measurement input, either by detection with recognition or detection without recognition. Recognition in this case means we know in the first place to which track a particular observation belongs. Foreground segmentation and optical flow are two methods of obtaining measurement input without recognizing the tracked object. Those algorithms work by detecting moving pixels without knowing the identity of the tracked object, and the detected foreground blob such as from [5] may contain more than a single tracked object. The detected foreground blob must be associated with a specific track in order to know the identity of the object. Multiple hypothesis approach [6] can be employed to optimize the association process so that no merge and split cases are considered. On the other hand, observations can be obtained by recognizing the object using feature-based and template-based recognition. We have pre-stored data of the tracked object where we will find it in the later frames. Therefore, the detected observation is already associated with that

particular track. During the initial stage of tracking, a new track can be initialized by the user or by using non-recognition-based detection. The features of the tracked object are then learned for recognizing it in later frames.

Since illumination change is hard to model with a single colour model, we implemented two colour models for PBOD—RGB and HSV—where the latter is heavily used during illumination change. For histogram matching, we explored five histogram similarity modelling: correlation [7], chi squared [8], intersection [9], Bhattacharyya [10] and Poisson distribution [11]. We reviewed and compared two schemes for PBOD [11], one deterministic and the other probabilistic. The former method relies on threshold-based decision making while the latter method implements maximum a posterior and Bayes risk for decision-making. Both algorithms share the same basic structure but differ in whether patch matching and smoothing procedures are deterministic or probabilistic. Details of the algorithms are explained in the methodology section. The main novelties of PBOD are (1) the fusion between a feature and template-based approach; (2) modelling histogram similarity by Poisson distribution; (3) probabilistically adjusted patch position and size to fit the tracked object completely. Finally, the output patch is fed into any filter-based tracker as the measurement input.

This paper is organized into 6 sections. A literature review will be presented in Section 2. Details of the algorithms are fully explained in Section 3, where each subsection explains in detail the methods to obtain points of interest, generate candidate patches, find the best patch, perform position alignment and adjust the patch's size. In Section 4, the pseudo-codes of both deterministic and probabilistic PBOD are given for more clarity. Simulation results and discussions are presented in Section 5. The conclusions are given in the last section to emphasize the performance difference.

2. Literature Review

Since we are focusing on obtaining an observation through recognition, the two most common methods of object recognition are the template- and feature-based methods. The feature-based approach usually recognizes the object by obtaining a match based on the feature descriptor. The template-based approach uses the shape or a collection of pixels in finding a match. The major trade-offs of both approaches are the distinctiveness property and the generalization property. Most feature-based approaches have high distinctiveness property but low in generalization property. This explains why feature-based approaches do not work well for blurred images but perform exceptionally well on rich textured objects. On the other hand, the template-based approach has low distinctiveness property but is very good in generalizes the object detection. Even blurred and non-rigid objects can still be recognized reliably.

Template-based recognition is an approach that requires a database or a collection of possible templates to be built before any matching can be performed. This method is used in many license plate recognition systems [12–14]. Templates of the characters are built under several viewing conditions before matching is performed. Another popular template-based system is human recognition such as the work by Hsieh *et al.* [15]. The authors divided the frames into nine sections where each section has a different database of the human silhouette. It reduced processing time significantly by anticipating the human shape at the selected viewing angle. This clever approach allows the system templates

to capture all possible transformations in a smaller database resulting in acceptable processing time. Sometimes, the templates or databases are obtained by extensive training even to the extent of 80 million images [16]. Simple template-based recognition is demonstrated by Bradski and Kaehler in [7], where they slide a single template across the whole image. The main advantage of the template-based approach is that it manages to recognize the object under moderate deformation, blurring and illumination change, all of which are common in video applications. An example template matching technique built for video application is the system by Cole *et al.* [17], where they used an adaptive scaling technique to reduce computation burden. The authors also reduce the number of matching candidates by segmenting the database into several classes and further breaking it down within a class into several groups. Yan *et al.* [18] then proposed a sub-template mean-shift method. m -best templates were chosen as the candidate based on their distance from the tracked object. This spatial distance also signified the a priori knowledge, which a Gaussian-based voting is employed to select the best template.

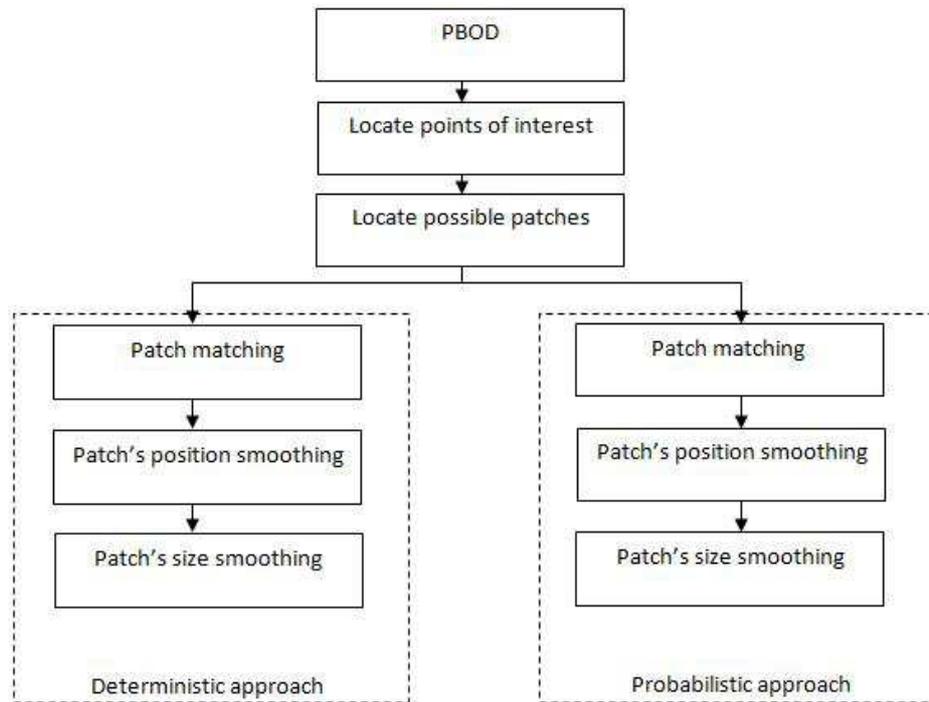
One of the most cited paper regarding point-based detector is Scale Invariant Feature Transform (SIFT) [19], which generates robust features that works exceptionally well even for problems of rotation, scaling and moderate illumination change. Ke and Suthankar [20] improved SIFT's feature distinctiveness property by applying principal components analysis. Later, Burghouts and Geusebroek [21] fused SIFT with a colour invariance [22] algorithm to achieve robustness to illumination changes. Their algorithm transformed the image into colour invariant forms, which they divided into **E**, **W**, **C** and **H** colour invariants before applying SIFT. Performance evaluations of SIFT and its variants are explained in depth by Mikolajczyk and Schmid [23]. They also introduced their own algorithm, GLoH, which improved SIFT by incorporating more spatial properties during histogram accumulation. In 2004, Ledwich and Williams [24] proposed a simplified SIFT feature for reducing computational burden. The author claimed that a significant gain in processing speed is obtained at minimal accuracy loss by utilizing structural similarity to reduce the number of keypoints generated. Moreover, Bay *et al.* [25] introduced Speeded Up Robust Features (SURF), which claims similar detection performance to SIFT but requires less computational burden. The underlying principals used in SURF are the combination of integral image and Hessian matrix based point descriptors.

Another method of obtaining tracking observation is by applying a histogram-based method. Histogram-based tracking algorithms [3,26] have been applied successfully to non-rigid objects because the matching is done based on the statistics of a group of pixels. A multilevel thresholding of the histogram data is used by Chen *et al.* [27] to improve the detection robustness to illumination changes and spurious infrared noise. The most popular histogram-based tracker is a mean shift algorithm [28] where the next location is predicted based on the input of histogram backprojection via the mean shift algorithm. Bradski introduced CAMSHIFT [26], which integrates scalability into the mean shift algorithm, thus allowing the tracked object to have variable size. A kernel-based tracker that utilizes the Epanechnikov kernel profile has been introduced by Comaniciu *et al.* [3]. This approach puts more emphasize on pixels that are closer to the anchor pixel and less weight on distant pixels for accumulating the histogram's bin values. They also apply the Bhattacharyya distance [10] for measuring similarity between two histograms.

In general, a kernel-based algorithm performs well for single object tracking [3,29,30]. However, as the scene becomes crowded and more objects need to be tracked, the algorithms start to falter, especially during occlusion, poor segmentation and inability to segment touching detections. The works by Namboodiri *et al.* [31] and Peng *et al.* [32] attempted to solve the problem of occlusion. The algorithm of Namboodiri *et al.* tweaks the localization of the mean shift by applying both forward and reverse methods so that it converges to the true modes. They also add scalability by utilizing SIFT's scale. However, this contradicts their assertion that their algorithm should require less processing power as it is a known fact that SIFT uses more processing power compared with the mean shift methods. The work by Peng *et al.* focuses on how to improve the updating method for the object model, for which they utilize a Kalman filter prediction method. The predicted object model is called the candidate model while the previous model is called the current model. Hypothesis testing is used to select the correct histogram model. In their paper, Leichter *et al.* [33] improved the kernel-based method by using multiple models for the object so that it tracks well under sudden changes of viewpoint. This method requires the user to initialize the object model in several views, which can be quite problematic. The main weakness of the mean shift algorithm is it depends on the proximity property where it is prone to failure when the object's movement is fast. The algorithm of Li *et al.* [30] approaches this problem by extending the search area based on their hexagon method. However, it is a brute force search that requires a significant amount of processing. Besides, the algorithm fails if the object moves fast enough such that it leaves their search region.

3. Patch-Based Observation Detection (PBOD): Deterministic and Probabilistic Approaches

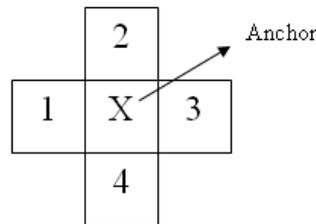
Both methods were first introduced in [11]. PBOD is built specifically for obtaining tracking observations in challenging surroundings and environment. This algorithm has a moderate distinctiveness property with better recognition accuracy for most applications. The distinctiveness property is the measure of the uniqueness of a descriptor, while generalization property is the ability to find a match in a noisy environment. A typical example is if the object is blurred, the descriptor will not be able to match the points of interest. However, if the generalization property is good, the algorithm can still find a match by generalizing the information of the object. The main challenge lies in recognizing the objects under illumination change, blurring effects, moderate deformation and non-rigid objects, low ambient illumination and objects with homogenous texture. We have developed two versions of the observation detector, the deterministic PBOD and the probabilistic PBOD. Both approaches share the same algorithm up to the possible patch generation. For the latter stage, the algorithms differ as indicated by the name where decisions are made deterministically or probabilistically. The main five components of the algorithm are shown in Figure 1.

Figure 1. Block diagram of deterministic and probabilistic PBODs.

3.1. Generate Points of Interest and Possible Patches

A point of interest is used to obtain the location to generate vector descriptors. These descriptors are matched between frames for building possible patches. The original location of the object or the original patch in the first frame is initialized by the user. The importance of this user defined patch is that it serves as the reference for building the statistical data used in matching and smoothing procedures and in particular, the reference histograms. Moreover, the size of the previous frame patch indicates the object's size. Let β_w and β_h denote the width and height of a bounding box or patch. For this section, both current $F_{R,G,B}^{t,x,y}$ and previous frames $F_{R,G,B}^{t-1,x,y}$ are transformed to grey scale space $(F_I^{t,x,y}, F_I^{t-1,x,y})$. Corner detectors as defined by Shi and Tomasi [34] are applied to find the possible points of interest. The threshold used in the Shi and Tomasi algorithm is around 0.01, which signifies the minimum eigenvalue threshold required for the point to be considered as a corner. This corner detector was selected because of its ability to generate points even under low ambient illumination and for low textured objects. For the previous frame, the points are generated inside the predefined patch only, while for the current frame, the points are generated for the entire image.

Then, vector descriptor $\mathbf{V}^{t,x,y}$ is used to match the points of interest between frames. Possible bounding boxes are generated at each corner where the vectors are matched. We use the RGB colour model, and each channel is treated separately. Therefore, there will be three sets of vectors for each point of interest, $\mathbf{V}_R^{t,x,y}, \mathbf{V}_G^{t,x,y}, \mathbf{V}_B^{t,x,y}$. The vectors are generated by finding the colour difference between the anchor pixel and its selected neighbouring pixels as shown in Figure 2. Let i denote the colour channel and $i \in \{R, G, B\}$.

Figure 2. Neighbourhood pattern used for vector generation.

$$\mathbf{V}_i^{t,x,y} = \{F_i^{t,x-1,y} - F_i^{t,x,y}, F_i^{t,x,y-1} - F_i^{t,x,y}, F_i^{t,x+1,y} - F_i^{t,x,y}, F_i^{t,x,y+1} - F_i^{t,x,y}\} \quad (1)$$

Each of the vector components is sorted from the lowest to the highest value. The reason for sorting is to account for the rotation of the object, while the use of colour differences allows the algorithm to find good points of interest even during an illumination change. This algorithm does not have rotation invariant abilities since all matchings are done based on the colour histogram. The reason for using 4-connected neighbourhood data instead of 8-connected neighbourhood is to produce as much as possible candidate patches in the early stage, which later will be filtered by the subsequent processes. Each vector from the initial frame is compared with each vector in the next frame. The decision rule $L_1^{t,x,y}$ for matching the vectors is shown in Equation (2) where the differences between each vector component are summed up, and the final value is obtained by combining all three channels' differences, *i.e.*, a Manhattan distance. Then it is compared with a predefined threshold, \mathcal{T}_1 , which was found from experiments to be optimal in the range 11 to 13. Let $L_2^{t,x,y}$ denote the label which is set to one when the vectors are matched and zero for unmatched vectors.

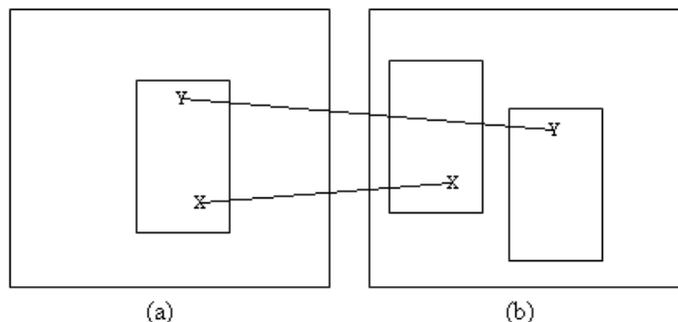
$$L_1^{t,x,y} = \sum_{\forall i} |\mathbf{V}_i^{t,x,y} - \mathbf{V}_i^{t-1,x,y}|, i \in \{R, G, B\} \quad (2)$$

$$L_2^{t,x,y} = \begin{cases} 1 & \text{if } L_1 < \mathcal{T}_1 \\ 0 & \text{if } L_1 \geq \mathcal{T}_1 \end{cases} \quad (3)$$

All the matched vectors are candidates for locations at which patches are built. Patches for the second frame are generated around the location of the matched vector in the first frame with respect to the original bounding box. Figure 3 shows an example of how the bounding box is generated. Initially, the size of the object is assumed to remain constant between frames.

A subsequent test for distinguishing overlapping patches is performed after all patches have been assigned location and size. This is performed due to different matched features possibly lying close to each other. If the difference is small, the patch should be smoothed out as one. This is done in order to reduce the calculation burden by reducing the number of patches. Moreover, most of the small differences occur because of “noise” in the patch generation process. The decision rule L_3 for determining overlapping patches is calculated as in Equation (4). Patch smoothing to combine all patches that lie close to each other is performed if the overlapping area O_a is more than $\mathcal{T}_2\%$ of the original patch size.

Figure 3. Examples of constructing new patches between the frames. The bounding boxes are aligned with respect to the matched vectors in the first frame. **(a)** First frame **(b)** Second frame.

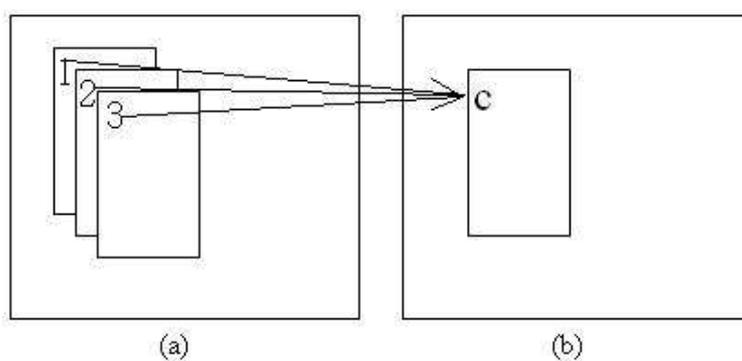


$$L_3 = \begin{cases} 1 & \text{if } O_a > (\mathcal{T}_2 \times \beta_w \times \beta_h) \\ 0 & \text{if } O_a \leq (\mathcal{T}_2 \times \beta_w \times \beta_h) \end{cases} \quad (4)$$

The new combined patch location, (\bar{x}, \bar{y}) is the average of the corresponding centers of the overlapping patches as shown in Figure 4 where N_L is the number of L_3 s detected. Comprehensive pseudo-code of both points of interest and possible patches generation are given in Algorithm 1.

$$\beta^{\bar{x}, \bar{y}} = \left(\frac{1}{N_L} \sum_{i=1}^{N_L} \beta^{x_i}, \frac{1}{N_L} \sum_{i=1}^{N_L} \beta^{y_i} \right) \quad (5)$$

Figure 4. Example of several patches combination. **(a)** Original patches **(b)** Combined patch.



Algorithm 1 Generate points of interest and possible patches

Require: Points of interest

- (a) Obtain user input bounding box during the first frame
- (b) Convert previous and current frames to grey scale for each cycle
- (c) Perform Smoothing to both frames with Gaussian convolution
- (d) Find Laplacian for each frame

(e) Obtain the matrix $\begin{bmatrix} F_{I_{xx}}^{t,x,y} & F_{I_{xy}}^{t,x,y} \\ F_{I_{yx}}^{t,x,y} & F_{I_{yy}}^{t,x,y} \end{bmatrix}$ for each pixel

(f) **If** min eigenvalue $> \mathcal{T}$
 accept the point as possible POI

Else

 reject the point

End If

Require: tentative patches

- (a) Each channel of RGB is treated separately
- (b) Built vector descriptors, $\mathbf{V}_i^{t,x,y}$ for both frames
- (c) **For** i, \dots , total POI **Do**

$$L_1 = \sum_{\forall i} |\mathbf{V}_i^{t,x,y} - \mathbf{V}_i^{t-1,x,y}|$$

End For

(d) Obtain L_2 to decide either to accept the POI or not

If $L_1 < \mathcal{T}_1$

$$L_2^{t,x,y} = 1$$

Else

$$L_2^{t,x,y} = 0$$

End If

(e) Construct a patch at each point where $L_2 = 1$

The original size of each patch is similar with the final size of the previous frame

(d) Smooth out overlapping patches by combining redundant patches

If $O_a > (\mathcal{T}_2 \times \beta_w \times \beta_h)$

 combine the patches by finding the mean of the patches

Else

 remain as it is

End If

3.2. Patch Matching

Patch matching is performed to find the patch where the object most likely resides. The match is done by comparing the histograms of previous and current frame patches. Two colour models are considered, *i.e.*, RGB and HSV. For the case of constant illumination, the RGB colour model gives better histogram comparison than does HSV. The RGB colour model is better due to its more distinctive feature when there is no illumination change. If illumination change occurs, the hue channel from the HSV colour model

gives better comparison performance. This is because the hue channel is more stable under moderate illumination change even though its distinctive property degrades. Another reason for choosing HSV colour model is due its simplicity, which results in low computational burden as compared with other colour invariant models such as [22,35]. For RGB colour space, a 3-dimensional histogram is built for each patch while a 1-dimensional histogram is built for the hue channel. The conversions from RGB model to HSV model are given by Tsai [36].

$$V = \frac{1}{3}(R + G + B) \quad (6)$$

$$S = 1 - \frac{3}{R + G + B} \min(R, G, B) \quad (7)$$

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360^\circ - \theta & \text{if } B > G \end{cases} \quad (8)$$

where

$$\theta = \cos^{-1} \left(\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \right) \quad (9)$$

3.2.1. Deterministic Approach

After the number of patches has been finalized, histogram correlation (\mathcal{D}_C) between current frame patches and previous frame patch is used to identify the object. The test is divided into two levels, where the first level is used to obtain the match under normal illumination, while the second-level test is initiated when an illumination change is detected. The first-level test depends on RGB colour space while a 1-dimensional hue histogram is used for the second level. Let N_b be the number of histogram bins in one dimension while previous and current frames' histograms are denoted by n and m respectively. m_i is the value of i^{th} bin of the current frame histogram where each histogram is normalized first before matching them. The output range is $[-1, 1]$, where 1 indicates a perfect match while -1 signifies a total mismatch. If \mathcal{D}_C is zero, it signifies a very low correlation value that indicates an illumination change has occurred or no match is found.

For a 1-dimensional histogram:

$$\mathcal{D}_{C_1}(n, m) = \frac{\sum_{i=1}^{N_b} n_i m_i - \frac{\sum_{i=1}^{N_b} n_i \sum_{i=1}^{N_b} m_i}{N_b}}{\sqrt{(\sum_{i=1}^{N_b} (n_i)^2 - \mathbf{K}) \cdot (\sum_{i=1}^{N_b} (m_i)^2 - \mathbf{L})}} \quad (11)$$

where

$$\mathbf{K} = \frac{(\sum_{i=1}^{N_b} n_i)^2}{N_b}, \text{ and } \mathbf{L} = \frac{(\sum_{i=1}^{N_b} m_i)^2}{N_b} \quad (12)$$

and for a 3-dimensional histogram:

$$\mathcal{D}_{C_3}(n, m) = \frac{\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} n_{i,j,k} m_{i,j,k} - \frac{\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} n_{i,j,k} \sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} m_{i,j,k}}{3N_b}}{\sqrt{(\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} (n_{i,j,k})^2 - \mathbf{K}) \cdot (\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} (m_{i,j,k})^2 - \mathbf{L})}} \quad (13)$$

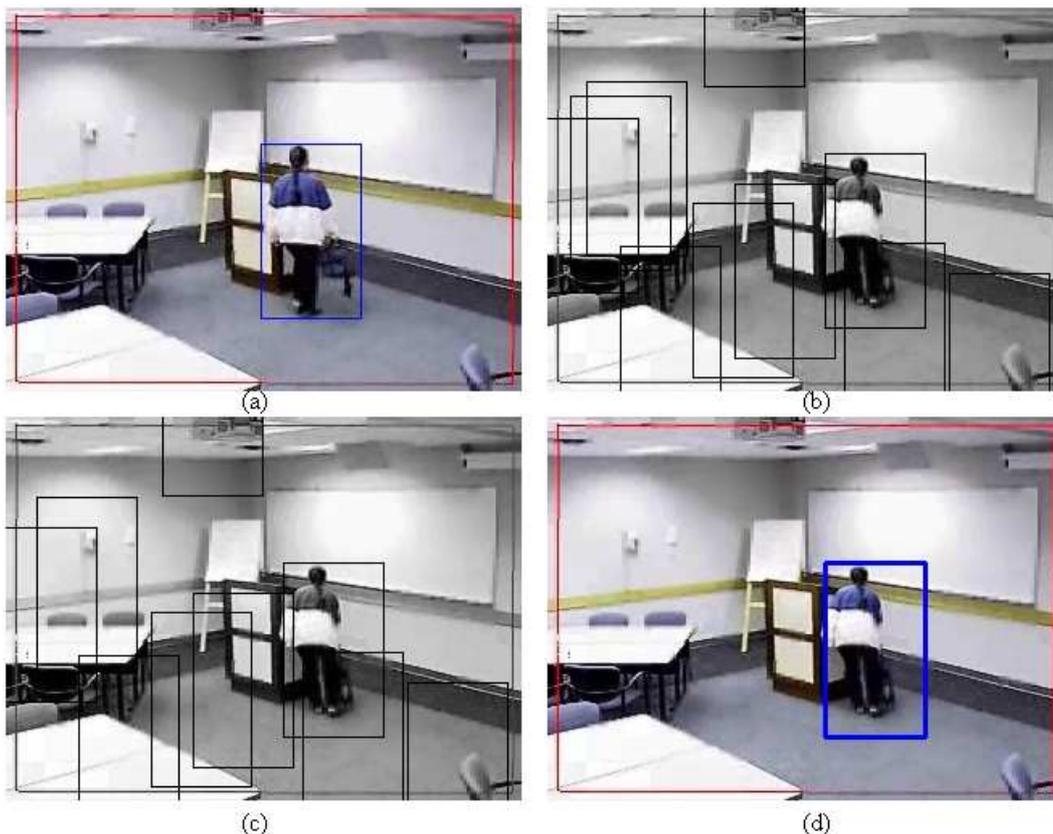
where

$$\mathbf{K} = \frac{(\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} n_{i,j,k})^2}{3N_b}, \text{ and } \mathbf{L} = \frac{(\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} m_{i,j,k})^2}{3N_b} \quad (14)$$

Since some of the matched vectors are found near the border of the image, certain patches may have some regions with components outside the frame. In this situation, we set out of bound components to be low (black), which consequently increases the probability of detecting occlusion. The patch with the highest correlation, β_1^{\max} , is taken as the candidate for the object location. However, $\beta_{1,R,G,B}^{\max}$ should exceed a predefined threshold \mathcal{T}_3 or otherwise the second-level test is initiated. The optimal value for \mathcal{T}_3 is found from extensive simulations to be around 0.73. Let ϵ_d be the indicator, which takes the value 1 if the first-level test is satisfied and 0 if the second-level test is initiated.

$$\epsilon_d = \begin{cases} 1 \text{ (1}^{st} \text{ level test) if } \beta_{1,R,G,B}^{\max} > \mathcal{T}_3 \\ 0 \text{ (2}^{nd} \text{ level test) if } \beta_{1,R,G,B}^{\max} \leq \mathcal{T}_3 \end{cases} \quad (15)$$

Figure 5. Procedures for selecting the right patch. (a) Original patch (b) Raw patches (c) Combined patches (d) Maximum correlation patch.



For the second-level test, both previous and current frames are transformed from RGB to HSV colour model. Only the hue channel is utilized where the illumination has changed due to the previous assumption. The reason for utilizing only hue information under illumination change is because of its stability compared with other colour information [7]. By using the same set of possible patches as in the first-level test, the hue histogram of each patch is obtained. Then, each correlation value is calculated using Equation (11) and the patch with the maximum correlation is taken as the candidate for the object

location. The maximum correlation is compared with the threshold value \mathcal{T}_4 in order to determine if the object still resides in the frame or not. Let L_4 represent the label of detecting the object. Figure 5 shows an example of selecting the right patch between two consecutive frames.

$$L_4 = \begin{cases} 1 \text{ (object is detected) if } \beta_{2,H}^{\max} > \mathcal{T}_4 \\ 0 \text{ (object have leaved the frame) if } \beta_{2,H}^{\max} \leq \mathcal{T}_4 \end{cases} \quad (16)$$

$$\beta_3 = \begin{cases} \beta_{2,H}^{\max} \text{ if } \epsilon_d = 0 \\ \beta_{1,R,G,B}^{\max} \text{ if } \epsilon_d = 1 \end{cases} \quad (17)$$

3.2.2. Probabilistic Approach

For the probabilistic approach, histogram matching is done by modelling the relationship between two histograms as a Poisson distribution as in Equations (18) and (19).

For a 1-dimensional histogram:

$$\mathcal{D}_{\mathcal{P}_1}(n, m) = \prod_{i=1}^{N_b} \left(\frac{\exp^{-n_i} n_i^{m_i}}{m_i!} \right) \quad (18)$$

and for a 3-dimensional histogram:

$$\mathcal{D}_{\mathcal{P}_3}(n, m) = \prod_{i=1}^{N_b} \prod_{j=1}^{N_b} \prod_{k=1}^{N_b} \left(\frac{\exp^{-n_{i,j,k}} n_{i,j,k}^{m_{i,j,k}}}{m_{i,j,k}!} \right) \quad (19)$$

A maximum likelihood approach is used to find the matched patch for both colour models. The likelihoods are modelled by Equations (18) and (19) where β_5 denotes the matched patch and \mathbf{x} represents the observation.

$$P_1(\mathbf{x}|\beta_4) = \begin{cases} \mathcal{D}_{\mathcal{P}_1}(n, m) \text{ for HSV colour model} \\ \mathcal{D}_{\mathcal{P}_3}(n, m) \text{ for RGB colour model} \end{cases} \quad (20)$$

$$\beta_5 = \underset{\forall \beta_4}{\operatorname{argmax}} P_1(\mathbf{x}|\beta_4) \quad (21)$$

There are two candidates for the most likely patch. The decision to choose the hue colour model over RGB is made by using a Neyman–Pearson hypothesis test [37]. Let $P(\mathbf{x}; H_0) = P_2(\beta_{4,R,G,B})$, $P(\mathbf{x}; H_1) = P_3(\beta_{4,H})$ and η_1 represent the threshold for the Neyman–Pearson hypothesis test. If the test favours H_0 , ϵ_p is initialized as one. On the other hand, if H_1 is chosen, ϵ_p is set equal to zero. The parameter ϵ_p indicates which colour space is used for position and size smoothing. The resultant patch β_6 from the test will be the final matched patch.

$$\text{NP}_1 = \frac{P_3(\beta_{4,H})}{P_2(\beta_{4,R,G,B})} > \eta_1 \quad (22)$$

$$\beta_6 = \begin{cases} \beta_{4,R,G,B} \text{ if } P_3(\beta_{4,H}) < \eta_1 P_2(\beta_{4,R,G,B}) \\ \beta_{4,H} \text{ if } P_3(\beta_{4,H}) \geq \eta_1 P_2(\beta_{4,R,G,B}) \end{cases} \quad (23)$$

3.3. Position Smoothing

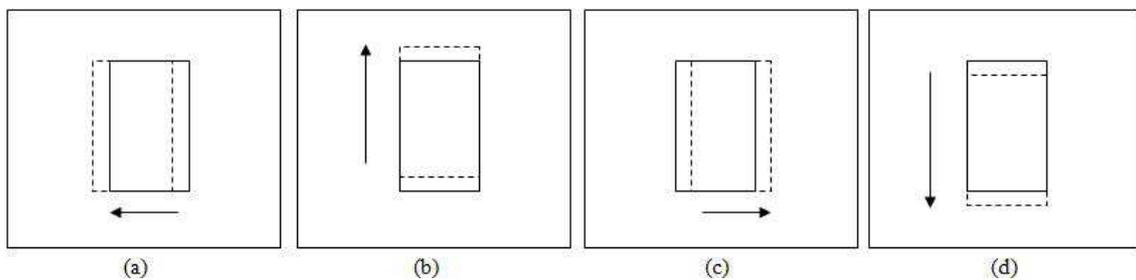
Position smoothing is used to adjust the patch's centroid to precisely fit the object's centroid. Sometimes, the calculated patch is slightly misaligned with the tracked object. This error is prevalent during illumination change and low ambient illumination. The adjustment is divided into two cases depending on the value of ϵ . The step size δ used for adjusting the patch translation is determined first. Let \mathcal{T}_5 denote a weight factor that takes values in $[0, 1]$.

$$\delta = \mathcal{T}_5(\min(\beta_w, \beta_h)) \quad (24)$$

3.3.1. Deterministic Approach

The translation test for adjusting the patch location is performed in four directions as shown in Figure 6, *i.e.*, leftward (β_7^a), upward (β_7^b), rightward (β_7^c) and downward (β_7^d). The solid line patch is the original position while the dashed line patch is the patch translated by a step size value. The statistical properties of these patches are retrieved depending on the ϵ_d value. For ϵ_d equal to 1, the RGB histograms for each patch are built and its correlation with the previous frame patch's histograms is calculated.

Figure 6. Patches coordination for location smoothing (a) Left side translation (b) Upward translation (c) Right side translation (d) Downward translation.



Every correlation (\mathcal{D}_c) of the four new patches ($\beta_7^a, \beta_7^b, \beta_7^c, \beta_7^d$) and the original patch (β_6) correlation ($\mathcal{D}_c^{\text{current}}$) are compared. The new patch location is selected based on the maximum correlation $\mathcal{D}_c^{\text{max}}$ among them. If the original patch correlation is the maximum, the position will remain the same. If any of the new patch's correlation is the maximum, the detected patch is shifted toward that corresponding direction. The new maximum correlation is reset when the patch is moved. The procedures are repeated until the maximum correlation among the new patches is less than the original patch correlation. $\mathcal{D}_c^{\text{max}}$ is stored for later usage during the shrinkage and expansion test.

$$\mathcal{D}_c^{\text{max}} = \max\{\mathcal{D}_c^a, \mathcal{D}_c^b, \mathcal{D}_c^c, \mathcal{D}_c^d\} \quad (25)$$

$$\beta_8 = \begin{cases} \beta_6 & \text{if } \mathcal{D}_c^{\text{max}} > \mathcal{D}_c^{\text{ori}} \\ \beta_7^{\text{max}} & \text{if } \mathcal{D}_c^{\text{max}} \leq \mathcal{D}_c^{\text{ori}} \end{cases} \quad (26)$$

For the case of ϵ_d equal to 0, only the hue channel histogram is generated instead of RGB histograms. The remaining steps follow the same procedure as before; the correlation is calculated and used for the

position smoothing comparison. All four new patch locations are tested, and the stopping criterion is when the maximum hue correlation among the new patches is less than the original patch correlation.

3.3.2. Probabilistic Approach

In the probabilistic approach, the same four new candidate patches are created for adjusting the patch position as shown in Figure 6, representing translations in four directions of the pivot patch (β_6)—leftward (β_9^a), upward (β_9^b), rightward (β_9^c) and downward (β_9^d). Histograms of each of the five patches, including the original position patch are obtained, and the maximum likelihood is used to find the new location. Likelihood is derived from the relationship between the previous and current frames' histogram as in the Equations (18) and (19). Let β_{10} denote the output of position smoothing.

$$P_4(\mathbf{x}|\beta_9) = \begin{cases} \mathcal{D}_{\mathcal{P}_1}(n, m) & \text{if } \epsilon_p = 0 \\ \mathcal{D}_{\mathcal{P}_3}(n, m) & \text{if } \epsilon_p = 1 \end{cases} \quad (27)$$

$$\beta_{10} = \underset{\forall \beta_9^i}{\operatorname{argmax}} P(\mathbf{x}|\beta_9), i \in \{a, b, c, d\} \quad (28)$$

For each iteration, the pivot position is reinitialized by letting $\beta_6 = \beta_{10}$, so that all four new translated patches for the next iteration are built around β_{10} . The algorithm is iterated until the estimated patch position remains the same as shown by the decision rule L_5 .

$$L_5 = \begin{cases} 0 & \text{(stop the iteration) if } \beta_{10} = \beta_6 \\ 1 & \text{(continue the iteration) if } \beta_{10} \neq \beta_6 \end{cases} \quad (29)$$

3.4. Size Smoothing

This section focuses on adjusting the size of the patch so that it provides a good fit to the tracked object. Generally, the apparent size of the object becomes bigger as it moves closer to the camera and smaller as it moves away. However, size increment and decrement between consecutive frames should not be very large. Based on this assumption, we limit the scale change for size smoothing by at most a factor of $\frac{1}{2}$ between two consecutive frames. Figure 7 shows an example of applying size smoothing in PBOD. The algorithm is still divided into two sections, which depend on the ϵ value as the colour space and parameters used are different. Eight new patches with different sizes are utilized for the size smoothing test. The same δ used in position smoothing is applied to adjust the patch size. Four shrinkage and four expansion pattern patches are obtained by either subtracting from or adding to one of the patch corners by a step size value. Figure 8 shows the shrunk patches ($\beta_{11}^a, \beta_{11}^b, \beta_{11}^c, \beta_{11}^d$), while the expanded patches are shown in Figure 9 ($\beta_{12}^a, \beta_{12}^b, \beta_{12}^c, \beta_{12}^d$).

Figure 7. Sample output of position and size smoothing algorithm (a) Original object (b) Blue box: Output of patch matching, Red box: Output of position smoothing and Green box: Output of size smoothing.



Figure 8. Patterns for shrinkage patch (a) Left side shrinkage (b) Upper side shrinkage (c) Right side shrinkage (d) Lower side shrinkage.

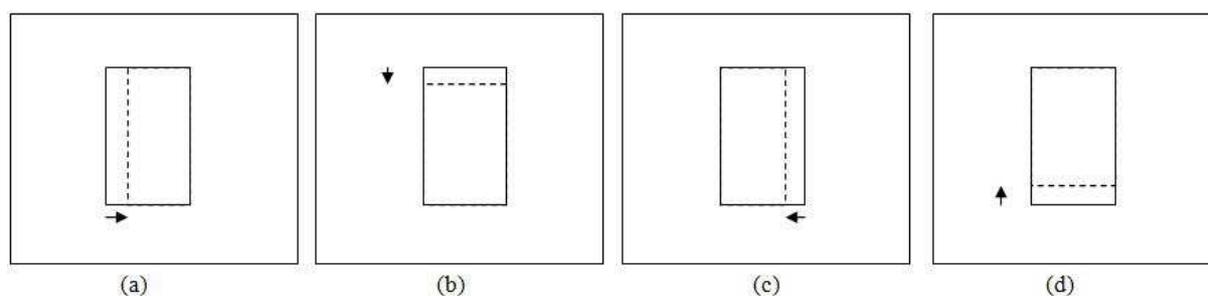
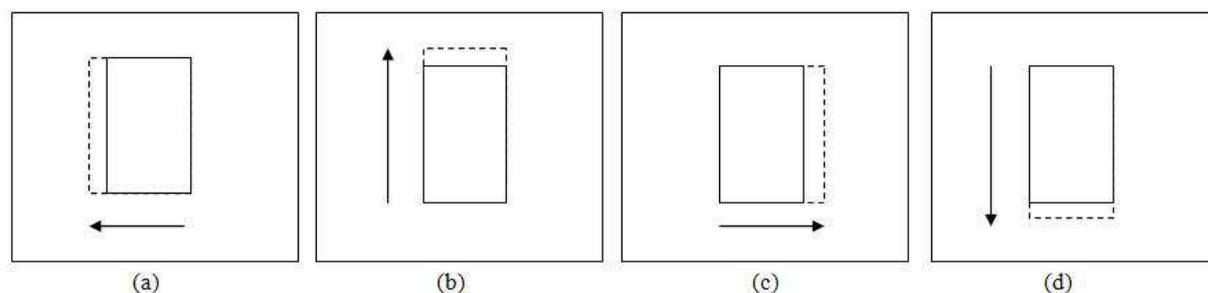


Figure 9. Patterns for expansion patch (a) Left side expansion (b) Upper side expansion (c) Right side expansion (d) Lower side expansion.



3.4.1. Deterministic Approach

We first consider the case where ϵ_d equals to one in which RGB channels are used for building the histogram. A parameter α is calculated as the weight in determining the size pattern.

$$\alpha = 0.1 \times (1 - \mathcal{D}_c^{\max}) \quad (30)$$

A test to determine the size pattern is performed to find out whether the object is expanding or shrinking. Here, only shrinkage patterns are considered. RGB histograms are generated for all new

shrinkage patterns. Then, the histogram's size is normalized before correlations between the new patches and the anchor patch β_8 are calculated. Let N_{β_1} and N_{β_2} denote the number of pixels inside patches from the previous and current frames respectively. Each histogram bin value H is adjusted by the ratio of N_{β_2} to N_{β_1} .

$$H^{\text{new}} = \left(\frac{N_{\beta_2}}{N_{\beta_1}} \right) H^{\text{old}} \quad (31)$$

The average correlation among the channels for each patch is calculated. The weighted correlation $\widehat{\mathcal{D}}_c$ between the new shrinkage patches are used to determine the size pattern. The weight \mathcal{T}_6 is used to find $\widehat{\mathcal{D}}_c$, which are comprised of $\mathcal{D}_c^{\text{max}}$ and the average of \mathcal{D}_c of the shrinkage patches.

$$\widehat{\mathcal{D}}_c = \mathcal{T}_6(\mathcal{D}_c^{\text{max}}) + (1 - \mathcal{T}_6)(\bar{\mathcal{D}}_c) \quad (32)$$

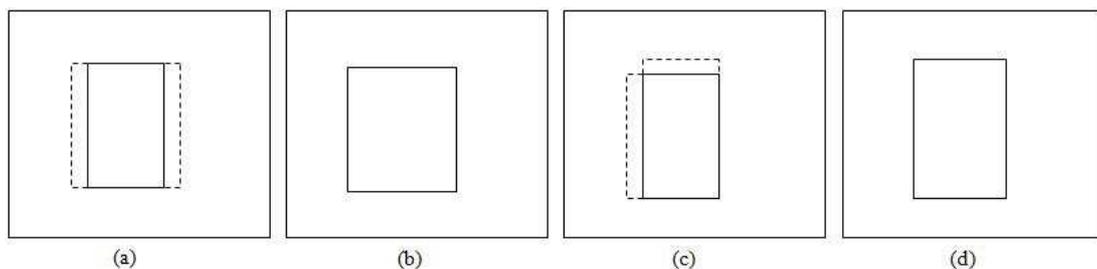
Then the weighted correlation is compared with the maximum correlation ($\mathcal{D}_{c\beta_8}$) from the location smoothing to determine the size pattern, L_6 .

$$L_6 = \begin{cases} -1 & \text{if } \widehat{\mathcal{D}}_c > \mathcal{D}_{c\beta_8} \\ 1 & \text{if } \widehat{\mathcal{D}}_c \leq \mathcal{D}_{c\beta_8} \end{cases} \quad (33)$$

For L_6 equal to one, RGB histograms of each of the new patches are generated as shown in Figure 9. The histograms are normalized before the correlations between the new patches and the previous frame patch are calculated. Each patch correlation is compared with $\mathcal{D}_c^{\text{max}}$. For each side, Equation (34) is used to decide whether expansion will occur or not. The size update is independent of the other expansion patches; any side can be expanded as long as it meets the requirement. Let L_7 be the decision label. A value of one denotes that the size is expanded while zero signifies that the size will remain constant. Thus, both left and right side patches can be updated at the same time as shown in Figure 10.

$$L_7 = \begin{cases} 0 & \text{if } \mathcal{D}_c^{\beta_i} < (\mathcal{D}_c^{\text{max}} - \alpha) \\ 1 & \text{if } \mathcal{D}_c^{\beta_i} \geq (\mathcal{D}_c^{\text{max}} - \alpha) \end{cases}, \beta_i \in \{\beta_{12}^a, \beta_{12}^b, \beta_{12}^c, \beta_{12}^d\} \quad (34)$$

Figure 10. Example of the patch expansion (a)(c) Original patch (b) Result if the right and left side expansion are true (d) Result if the left and upper side expansion are true.



Algorithm 2 Deterministic PBOD

Require: patch matching

- (a) For each patch, obtain \mathcal{D}_{C_3} based on RGB space
- (b) **If** $\beta_{1,R,G,B}^{\max} > \mathcal{T}_3$
 - $\epsilon_d = 1$
 - Else**
 - $\epsilon_d = 0$
 - End If**
- (c) **If** $\epsilon_d = 0$, start 2^{nd} level test
- (d) For 2^{nd} level test, obtain \mathcal{D}_{C_1} for each patch
- (e) Select $\beta_{2,H}^{\max}$ as the candidate patch

Ensure: position smoothing

- (a) Select colour space based on ϵ_d , either RGB or HSV
- (b) Determine the step size, δ
- (c) Construct the translated patch, $\beta_7^a, \beta_7^b, \beta_7^c, \beta_7^d$
- (d) Find maximum \mathcal{D}_C
- (e) **While** $\beta_8 = \beta_6$ **Do**
 - If** $\mathcal{D}_C^{\max} > \mathcal{D}_C^{\text{ori}}$
 - $\beta_8 = \beta_6$
 - Else**
 - $\beta_8 = \beta_7^{\max}$
 - End If**
- End While**

Ensure: size smoothing

- (a) Determine α as the size factor
 - (b) Obtain size pattern based on β_9 histograms either to use shrinkage or expansion pattern
 - (c) Normalize histograms size for fairer histogram matching
 - (d) **If** $\widehat{\mathcal{D}}_C > \mathcal{D}_{C\beta_8}$
 - Utilize shrinkage patch patterns
 - Else**
 - Utilize expansion patch patterns
 - End If**
 - (e) **Switch**
 - Case** (shrink)
 - If** $\mathcal{D}_C^i < (\mathcal{D}_C^{\max} - \alpha)$, $L_{13} = 0$ **Else** $L_{13} = 1$
 - Case** (expand)
 - If** $\mathcal{D}_C^j < (\mathcal{D}_C^{\max} + \alpha)$, $L_{14} = 0$ **Else** $L_{14} = 1$
 - End Switch**
 - (f) Reiterate the process until the patch has converged to a certain size or number of iteration has exceeded p cycles.
-

For the shrinkage pattern, the patches used are shown in Figure 9. The same steps used in the expansion pattern are applied but with a different decision rule L_8 as shown in Equation (35). A more stringent threshold is used in the shrinkage test to counter the homogenous texture problem. This is because the test will give good correlation even though the object size is not shrinking.

$$L_8 = \begin{cases} 0 & \text{if } \mathcal{D}_c^{\beta_j} < (\mathcal{D}_c^{\max} + \alpha) \\ 1 & \text{if } \mathcal{D}_c^{\beta_j} \geq (\mathcal{D}_c^{\max} + \alpha) \end{cases}, \beta_j \in \{\beta_{11}^a, \beta_{11}^b, \beta_{11}^c, \beta_{11}^d\} \quad (35)$$

For the case of ϵ_d equal to zero, the same set of algorithms are used but instead of RGB channels, only the hue channel is applied. The parameter α is replaced with α_1 and α_2 . Both parameters are predefined values, as it is hard to find good closed form expression for them due to the complexity of the scene when the illumination changes. α_1 is applied during the test for determining the size pattern, while α_2 is applied during the shrinkage and expansion test. The algorithm stops when there is no size change or the iteration has exceeded two times. Full pseudo-code for the deterministic approach is given in Algorithm 2.

3.4.2. Probabilistic Approach

For the probabilistic approach, β_{10} is used as the pivot point for creating all the new patches. A Bayesian approach is used to decide the final patch size β_{13} from among the nine patches, including the original patch β_{10} .

$$P_5(\beta_{11}|\mathbf{x}) = \frac{P_5(\mathbf{x}|\beta_i)P_5(\beta_i)}{P_5(\mathbf{x})}, \beta_i \in \{\beta_{10}, \beta_{11}^a, \dots, \beta_{11}^d, \beta_{12}^a, \dots, \beta_{12}^d\} \quad (36)$$

Since $P(\mathbf{x})$ is equal for all nine patches:

$$P_5(\beta_{11}|\mathbf{x}) \propto P_5(\mathbf{x}|\beta_i)P(\beta_i), \beta_i \in \{\beta_{10}, \beta_{11}^a, \dots, \beta_{11}^d, \beta_{12}^a, \dots, \beta_{12}^d\} \quad (37)$$

The value of ϵ_p determines what type of histogram is built. If ϵ_p equal to zero, a 1-dimensional hue histogram is used, while for ϵ_p equal to one, a 3-dimensional RGB histogram is applied. Before any comparison is performed, the histogram size must first be normalized. The normalization of the histogram size follows the same Equation (31) as the deterministic case. Once again, the histogram relationship between the previous and the current frame patches are modelled by a Poisson distribution.

$$P_5(\mathbf{x}|\beta) = \begin{cases} P_5(n, m) & \text{if } \epsilon_p = 0 \\ P_5(n, m) & \text{if } \epsilon_p = 1 \end{cases} \quad (38)$$

Two sets of prior probabilities are used. These depend on whether the size of the detected object inclines towards expansion or shrinkage. The selection of a suitable prior probability is very important, as the likelihood of shrinkage is usually large even when the object expands. Thus, we apply lower prior probabilities to shrinkage candidates if the size is increasing. In order to determine which set of the prior probabilities to use, again a Neyman–Pearson hypothesis test is implemented, where H_0 and H_1 represent the expansion and shrinkage hypotheses. Only eight candidate patches are used (four shrinkage

patches + four expansion patches) for this test where the same Poisson distribution as in Equation (38) is used. The maximum probability among the expansion patches represents the H_0 probability, while the maximum probability among the shrinkage patches represents the H_1 probability.

$$[t]P(\mathbf{x}; H_0) = \max_{\forall \beta^i} P_5(\mathbf{x}|\beta_i), \beta_i \in \{\beta_{12}^a, \dots, \beta_{12}^d\} \quad (39)$$

$$P(\mathbf{x}; H_1) = \max_{\forall \beta^j} P_5(\mathbf{x}|\beta_j), \beta_j \in \{\beta_{11}^a, \dots, \beta_{11}^d\} \quad (40)$$

Let η_2 be the threshold for the Neyman–Pearson test.

$$\text{NP}_2 = \frac{P(\mathbf{x}; H_1)}{P(\mathbf{x}; H_0)} > \eta_2 \quad (41)$$

$$P_5(\beta) = \begin{cases} P(\beta^{\text{expand}}) & \text{if } H_0 \text{ is true} \\ P(\beta^{\text{shrink}}) & \text{if } H_1 \text{ is true} \end{cases} \quad (42)$$

After the prior probability is obtained, each of the nine patches posterior ($P_5(\beta_{10}|\mathbf{x})$, $P_5(\beta_{11}^a|\mathbf{x})$, \dots , $P_5(\beta_{11}^d|\mathbf{x})$, $P_5(\beta_{12}^a|\mathbf{x})$, \dots , $P_5(\beta_{12}^d|\mathbf{x})$) is calculated. Each side of the bounding box can be expanded or shrunk independently based on the L_9 decision rule. Each side size is altered depending on whether the new posteriors exceed the original size posterior. L_9 equal to one indicates that the size is updated, while L_9 equal to zero indicates that the size remains constant.

$$L_9 = \begin{cases} 0 & \text{if } P_5(\beta_{10}|\mathbf{x}) \leq P_5(\beta_i|\mathbf{x}) \\ 1 & \text{if } P_5(\beta_{10}|\mathbf{x}) > P_5(\beta_i|\mathbf{x}) \end{cases}, \beta_i \in \{\beta_{11}^a, \dots, \beta_{11}^d, \beta_{12}^a, \dots, \beta_{12}^d\} \quad (43)$$

Probabilistic PBOD will follow the same rules as deterministic PBOD, which allows each side to be independently updated as shown in Figure 10. The iteration is terminated if no change is detected. Algorithm 3 denotes the pseudo-code of the probabilistic approach from patch matching to size smoothing.

4. Simulation Results and Discussion

The accuracy and effectiveness of PBOD were validated rigorously, in which the simulations are divided into three subsections:

1. Histogram matching performance
2. Deterministic and probabilistic PBOD
3. Probabilistic PBOD, Kernel tracker and SIFT-based tracker.

Algorithm 3 Probabilistic PBOD

Require: patch matching

- (a) Calculate $\mathcal{D}_{\mathcal{P}_1}$ and $\mathcal{D}_{\mathcal{P}_3}$ for all patches
- (b) Find β^{\max} for both $\mathcal{D}_{\mathcal{P}_1}$ and $\mathcal{D}_{\mathcal{P}_3}$
- (c) Apply Neyman–Pearson to decide between RGB and HSV colour space
 - If** H_1 is true
 - $\beta_6 = \underset{\forall \beta}{\operatorname{argmax}} \mathcal{D}_{\mathcal{P}_3}$
 - Else**
 - $\beta_6 = \underset{\forall \beta}{\operatorname{argmax}} \mathcal{D}_{\mathcal{P}_1}$
- End If**

Ensure: position smoothing

- (a) Select colour space based on ϵ_p
- (b) Determine the step size, δ
- (c) Construct the translated patch, $\beta_7^a, \beta_7^b, \beta_7^c, \beta_7^d$
- (d) For each patch, histogram matching is modelled by Poisson distribution
- (e) Apply maximum likelihood for position adjustment
- (f) **While** $\beta_{10} = \beta_6$ **Do**
 - $\beta_{10} = \underset{\forall \beta_9^i}{\operatorname{argmax}} P(\mathbf{x}|\beta_9)$
- End While**

Ensure: size smoothing

- (a) Obtain prior probability by using Neyman–Pearson test
 - (b) Obtain null hypothesis, H_0
 - $\max_{\forall \beta^i} P_5(\mathbf{x}|\beta_i), \beta_i \in \{\beta_{12}^a, \dots, \beta_{12}^d\}$
 - (c) Obtain alternative hypothesis, H_1
 - $\max_{\forall \beta^j} P_5(\mathbf{x}|\beta_j), \beta_j \in \{\beta_{11}^a, \dots, \beta_{11}^d\}$
 - (d) **If** H_1 is favoured
 - select 2^{nd} set of priors
 - Else**
 - select 1^{st} set of priors
 - End If**
 - (e) For each patch, obtain posterior probability by using Bayes risk
 - (f) Normalize all histograms size for fairer comparison
 - (g) **If** $P_5(\beta_{10}|\mathbf{x}) \leq P_5(\beta_i|\mathbf{x}), \beta_i \in \{\beta_{11}^a, \dots, \beta_{11}^d, \beta_{12}^a, \dots, \beta_{12}^d\}$
 - Size remain constant
 - Else**
 - size is updated based on selected side
 - End If**
 - (h) Reiterate the process until the patch has converged to a certain size or number of iteration has exceeded p cycles.
-

We have selected 150 image pairs from various videos from Youtube, which contain challenging scenes between two consecutive frames. Some of the challenges that reduce accuracy and precision of the tracker are illumination changes, shadows, non-rigid object, blur and partial occlusion. The size of the frame varies from 320×240 to 960×720 . The target object is not just a human, but also includes book, animal, ball and many more for both indoor and outdoor environments. However, only one object is tracked each time, since we limit the algorithm to single object tracking. Our tracked object varies in size from frame to frame and from video to video, in which the smallest size is 30×6 , and the largest size is 341×365 . Generally, a bigger tracked object will tend to perform better due to the smaller number of candidate patches after patch smoothing. They also tend to overlap with the ground truth after the patch matching process, which later will be fine-tuned by position and size smoothing. For a small size object, the possibility of overlapping is smaller compared with the bigger object, which diminishes the advantage of having position and size smoothing processes. Table 1 shows the parameters used by our algorithm. For both likelihood tests, we found out that 0.0001 gives the best result. In order to get a decent result, it is recommended that η_1 and η_2 should be within $[0.00005, 0.0005]$ based on our repeated simulations, while \mathcal{T}_1 and \mathcal{T}_2 between 0.6 to 0.75 will give good results. \mathcal{T}_3 is used to determine either to continue the test to HSV space or just stop at RGB space, while \mathcal{T}_4 is the threshold to indicate that the object is already out of the frame. So, correlation value of 0.7 and above will give good results for both tests, where less stringent value will favour RGB space while a more stringent value will favour HSV space. Step size will be determined by \mathcal{T}_5 and we use 0.1 scale of the tracked object's size. Smaller step size will give better accuracy but total number of iterations will be bigger and *vice versa*. \mathcal{T}_6 is used to weight the contribution of maximum correlation and average correlation for size smoothing procedures in deterministic PBOD. We reduced the effect of dependency on maximum correlation alone, as sometimes it may be obtained from noisy or blur patch by adding the averaging components. 0.5 is chosen as it gives good balance between both components. We also analyze the effect of the histogram's size ($N_b = 25, 50, 75$) where various methods of histogram matching are used for patch matching as shown in Table 2.

Table 1. Parameters used by our algorithm.

Parameter	Value
η_1	0.0001
η_2	0.0001
\mathcal{T}_1	0.6
\mathcal{T}_2	0.6
\mathcal{T}_3	0.7
\mathcal{T}_4	0.7
\mathcal{T}_5	0.1
\mathcal{T}_6	0.5
N_b	50

Table 2. Comparison of the average distance error among histogram matching methods: A: correlation, B: chi-square, C: intersection, D: Bhattacharyya and E: Poisson for various histogram's size.

Method	A	B	C	D	E
Total no. of histogram's bin: 25 bins					
Average \mathcal{R}_D (%)	26.51	22.35	26.88	22.37	19.16
Total no. of histogram's bin: 50 bins					
Average \mathcal{R}_D (%)	28.62	19.93	20.87	20.17	16.36
Total no. of histogram's bin: 75 bins					
Average \mathcal{R}_D (%)	21.11	17.27	17.84	18.04	15.02

The algorithm performance is measured by calculating the Euclidean distance \mathcal{E} between the centroid (Ω_{sim}) of the simulation result and the manually determined ground truth (Ω_{truth}) of the detected object.

$$\mathcal{E} = \sqrt{(\Omega_{sim}^x - \Omega_{truth}^x)^2 + (\Omega_{sim}^y - \Omega_{truth}^y)^2} \quad (44)$$

In terms of average processing speed, the method by Yan *et al.* performs the best with one frame per second (fps), followed by deterministic PBOD, probabilistic PBOD and SIFT-based tracker with 0.27 fps, 0.23 fps and 0.22 fps respectively. The method by Yan *et al.* has the lowest computational burden, since the search scope is limited to the neighbourhood data only, while the other three methods search the whole frames selectively by finding possible keypoints. Computational time for SIFT-based approach is slower because of the complex histogram of gradient and the longer descriptor used for matching as compared with PBOD, which uses simple histogram matching. Table 3 shows the computational limit based on Big-O notation for all four methods.

Table 3. Big-O notation for the algorithms.

Method	T(n)
Deterministic PBOD	1 $O(n^3)$ + 7 $O(n^2)$
Probabilistic PBOD	2 $O(n^4)$
Yan <i>et al.</i>	3 $O(n^2)$ + 8 $O(n)$
SIFT based approach	$O((2n)^2)$ + 2 $O(n^2)$

4.1. Histogram Matching

In this subsection, we demonstrated that the most appropriate histogram matching for PBOD is by using Poisson modelling. Partial of the probabilistic PBOD, which is without position and size

smoothing is used to validate the best scheme for histogram matching. Five methods have been tested on 150 image pairs for various image conditions, including illumination changes, shadow, non-rigid and homogenous texture object. Those five methods are:

1. Correlation distance, \mathcal{D}_C .
2. Poisson distance, \mathcal{D}_P .
3. Chi square distance, \mathcal{D}_{χ^2} .
4. Intersection distance, \mathcal{D}_I .
5. Bhattacharyya distance, \mathcal{D}_B .

Correlation distance is based on Bradski and Kaehlerv [7] as in Equations (11) and (13), which have been applied in deterministic PBOD. Poisson distance has been implemented in probabilistic PBOD as in Equations (18) and (19), which is based on Zulkifley and Moran [11]. Chi-square distance is taken from Schiele and Crowley [8], where the zero indicates a perfect match. Similar to Section 3.2.1, let n and m denote previous and current frames' histograms respectively. m_i is the value of i^{th} bin of the current frame histogram where each histogram is normalized first before matching process. After histogram normalization, the range for both n_i and m_i is $[0, 1]$. In this paper, we have tested three N_b values, which are 25, 50 and 75.

For a 1-dimensional histogram:

$$\mathcal{D}_{\chi_1^2}(n, m) = \sum_{i=1}^{N_b} \left[\frac{(n_i - m_i)^2}{n_i + m_i} \right] \quad (45)$$

and for a 3-dimensional histogram:

$$\mathcal{D}_{\chi_3^2}(n, m) = \sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} \left[\frac{(n_{i,j,k} - m_{i,j,k})^2}{n_{i,j,k} + m_{i,j,k}} \right] \quad (46)$$

Intersection distance is based on the work by Swain and Ballard [9] with the output range of $[0,1]$. It is obtained after normalization process with respect to the total number of pixels in the patch. The method accumulates the minimum bin values between the two histograms as shown in Equations (47) and (48).

For a 1-dimensional histogram:

$$\mathcal{D}_{I_1}(n, m) = \sum_{i=1}^{N_b} \min(n_i, m_i) \quad (47)$$

and for a 3-dimensional histogram:

$$\mathcal{D}_{I_3}(n, m) = \sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} \min(n_{i,j,k}, m_{i,j,k}) \quad (48)$$

Early formulation of Bhattacharyya distance can be traced back to [10], while current implementation is derived from Bradski and Kaehler [7]. The highest distance, one, indicates a total mismatch, while the

lowest score, zero, shows the perfect match.

For a 1-dimensional histogram:

$$\mathcal{D}_{\mathcal{B}_1}(n, m) = \sqrt{1 - \sum_{i=1}^{N_b} \left(\frac{\sqrt{n_i \cdot m_i}}{\sqrt{\sum_{i=1}^{N_b} n_i \cdot \sum_{i=1}^{N_b} m_i}} \right)} \quad (49)$$

and for a 3-dimensional histogram:

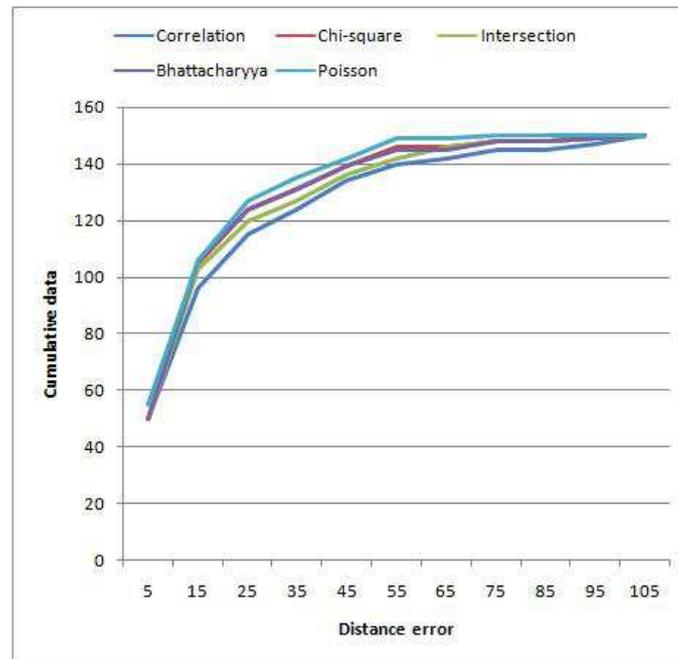
$$\mathcal{D}_{\mathcal{B}_3}(n, m) = \sqrt{1 - \sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} \left(\frac{\sqrt{n_{i,j,k} \cdot m_{i,j,k}}}{\sqrt{\sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} n_{i,j,k} \cdot \sum_{i=1}^{N_b} \sum_{j=1}^{N_b} \sum_{k=1}^{N_b} m_{i,j,k}}} \right)} \quad (50)$$

Table 4. Centroid distance for histogram matching methods: A: correlation, B: chi-square, C: intersection, D: Bhattacharyya and E: Poisson.

Distance Error (Pixel)	Number of Image Pairs				
	A	B	C	D	E
0–9	50	50	50	50	55
10–19	46	55	53	56	51
20–29	19	19	17	18	21
30–39	9	7	7	7	8
40–49	10	8	9	8	7
50–59	6	7	6	6	7
60–69	2	0	4	0	0
70–79	3	2	2	3	1
80–89	0	0	0	0	0
90–99	2	1	1	1	0
>99	3	1	1	1	0

Table 4 shows the error distance among the five methods of histogram matching. The centroid of the output patches is used as the reference point to calculate the error. We ran the algorithm for both colour models (RGB and HSV), and the minimum distance is taken since the algorithm for colour space selection is similar for all the methods. The results show that the most suitable histogram matching for PBOD is achieved by using Poisson modelling. Using the Poisson distance, 36.67% of the image pairs achieved an error distance of less than 10 pixels, while the rest just managed to get 33.33%. The most unsuitable method is the correlation distance, which has three wrong detections with error more than 100 pixels. Figure 11 shows the cumulative distribution of the error distance among the matching methods. This again proved that Poisson modelling is the best for PBOD with the steepest curve followed by Bhattacharyya, Chi-square, Intersection and correlation method.

Figure 11. Cumulative distribution of error distance among the histogram matching methods.



Distance error relative to the patch size \mathcal{R}_D is also calculated to show the magnitude of the distance error compared with the object size. The average distance error among the methods is shown in Table 2. We verify the matching test by using three sizes of the histogram, which are 25, 50 and 75 bins. For every bin's size, Poisson test results in the lowest average distance error, while correlation method performs the worst with the highest error for each size. Intersection, chi-square and Bhattacharyya test obtains almost similar error given the same histogram's size. The table also reveals that the error becomes smaller as the total number of bins increases except for the correlation method. The reason is that the distinguishing factor between two similar histograms is widened as the number of bin increases, which leads to more unique feature. Thus, the histogram matching tests have more probabilities of finding the right patch. However, there is no free lunch since the computational time increases as the number of bins increase. From our experience, 50 bins are the most suitable setup given the tradeoff between computational time and accuracy, since the algorithm improves by one percent only for an increment of 25 histogram's bin.

$$\mathcal{R}_D = \frac{\mathcal{D}}{\max\{\beta_w^{\text{new}}, \beta_h^{\text{new}}\}} \times 100\% \quad (51)$$

4.2. Deterministic and Probabilistic PBOD

This subsection is intended to prove that probabilistic PBOD performs better than deterministic PBOD. 120 image pairs are used to verify the performance difference. Again, Equation (44) is used to calculate the error distance of the centroid. Table 5 shows the comparison of error distance between probabilistic and deterministic PBOD. Probabilistic PBOD manages to obtain 48.33% detection with less than 10 pixels error while the deterministic approach manages only 42.50%. Mostly, probabilistic PBOD obtains a better result during illumination changes. Deterministic PBOD uses the hierarchical

approach where RGB space is searched first and if the resultant output is within the accepted region, no HSV space will be searched. On the other hand, probabilistic PBOD searches both spaces in parallel, where the best output is selected. If deterministic PBOD obtains a reasonably good result in RGB space, it will not search HSV space where it might obtain better matching. Thus, statistical PBOD manages to obtain better matching in challenging scenes.

Table 5. Comparison of the centroid distance between deterministic and probabilistic PBOD.

Distance error (pixel)	Number of image pairs	
	Deterministic	Probabilistic
0–9	51	58
10–19	31	43
20–29	14	12
30–39	4	2
40–49	2	2
50–59	4	2
60–69	0	1
70–79	2	0
80–89	2	0
90–99	1	0
>99	9	0

Moreover, no error of more than 99 pixels has been observed for the probabilistic PBOD while there are nine image pairs for the deterministic PBOD. The average distance error relative to the output patch size is given in Table 6, which is derived via Equation (51). Figure 12 shows the cumulative distribution of the error between both methods. It shows that 50% of the detections for probabilistic PBOD have less than 11 pixels error distance, while deterministic PBOD requires at least 14 pixels error. Another reason for probabilistic PBOD better performance is due to Poisson modelling where we proved before that it performs the best for PBOD as compared with correlation matching in deterministic PBOD. Poisson matching performs better during a blur case because it does not punish severely neighbourhood shift in histogram's value. The procedures of position and size smoothing of probabilistic PBOD are in the sequential manner where the first step of the patch's direction is very crucial, while probabilistic PBOD calculates the whole search space first before going into any direction. The main downside of searching the whole possible space is longer computational time. Figure 13 is an example of performance difference between probabilistic and deterministic PBOD under sudden illumination change. Figure 14 shows that both PBOD schemes work well in detecting the tracked object even if the object appearance is very blurred.

Table 6. Comparison of the average distance error between deterministic and probabilistic PBOD.

Method	Deterministic PBOD	Probabilistic PBOD
Average \mathcal{R}_D (%)	21.03	10.03

Figure 12. Cumulative distribution of error distance between probabilistic and deterministic PBOD.

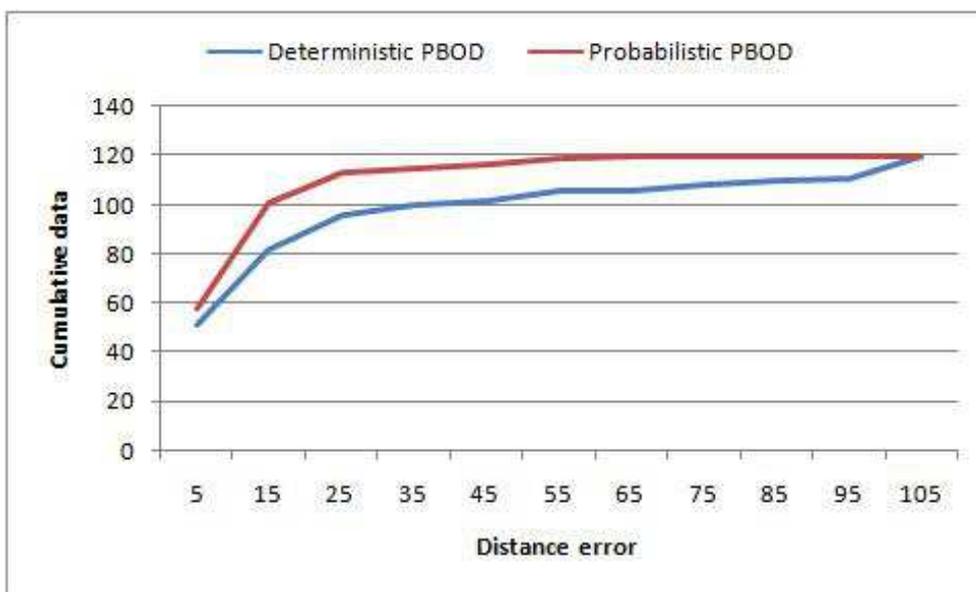


Figure 13. Deterministic and probabilistic PBOD under illumination change: (a) Deterministic PBOD (b) Probabilistic PBOD.

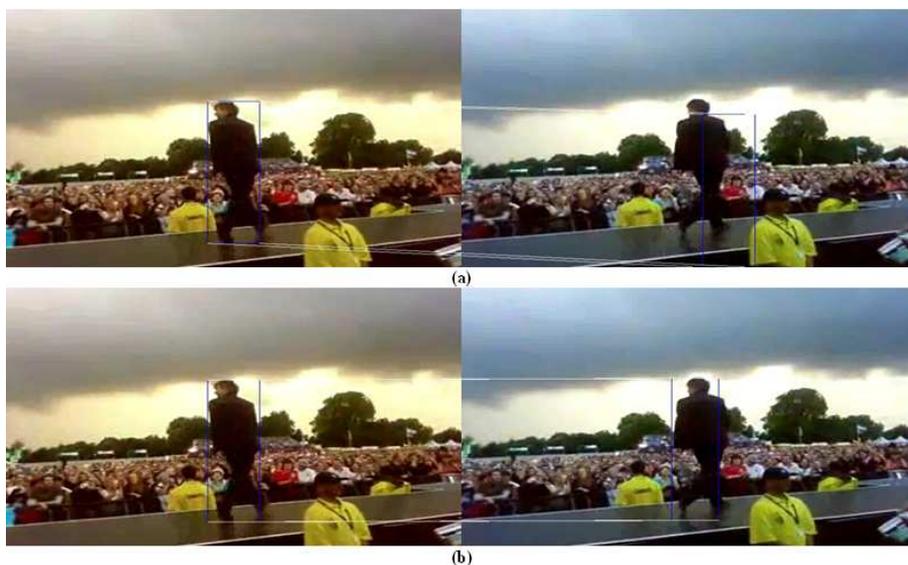
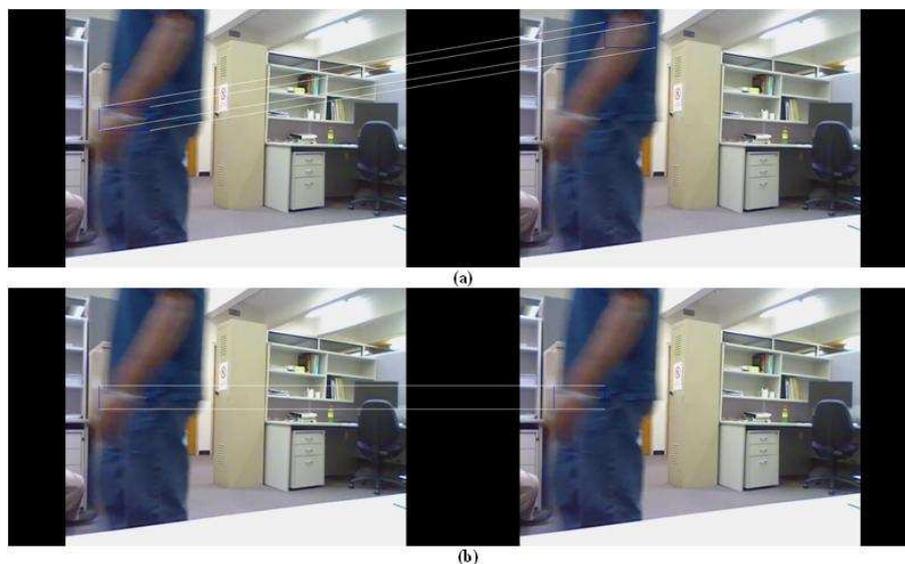


Figure 14. Deterministic and probabilistic PBOD for blur object: (a) Deterministic PBOD (b) Probabilistic PBOD



4.3. Probabilistic PBOD, Kernel Tracker and SIFT-based Tracker

In this section, we compare probabilistic PBOD with two other trackers, *i.e.*, kernel tracker and SIFT-based tracker. We have tested the algorithms on 120 pairs of video sequences that contain moving objects in various frame sizes. Kernel tracker represents the mean shift approach while SIFT-based tracker represents the feature-based tracker approach. Kernel tracker by Yan *et al.* [18] is chosen as the benchmark due to close similarity to our approach. The method is based on several possible sub-templates, which is built optimally before template voting is performed to select the best match. Their method also performs size smoothing deterministically by varying the template size on the scale of [0.95, 1.05]. On the other hand, SIFT-based tracker is built to compare our algorithm to feature-based tracker by tracking matched point in the consecutive frames. The four corners of PBOD bounding box are used as the reference points for centroid calculation. The centroid for SIFT-based tracker is generated by constructing a bounding box, which uses the extreme points in four directions as shown in Figure 15. Then, the generated bounding box corners are used for the centroid calculation.

Figure 15. Generating centroid for the SIFT-based tracker (a) Matched points of interest (b) Constructing a bounding box (c) Centroid location is indicated by the blue star.

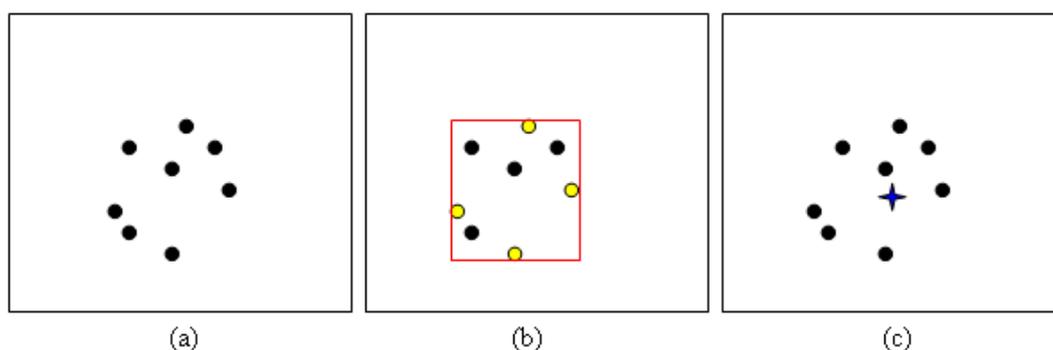


Table 7 shows the distance error analysis among the methods. Probabilistic PBOD performance is the best with 43.3% of the detected object centroid having less than 10 pixels distance from the ground truth centroid. Kernel tracker manages to obtain 37.5% detection with less than 10 pixels distance error while the worst is SIFT-based tracker with just 33.3% detection. Average distance error relative to the patch size (Equation (51)) is given in Table 8, where the error rate of probabilistic PBOD is just 13.95% while kernel tracker has 15.88% and SIFT approach has 35.93%. The cumulative error distance is given in the Figure 16, where 50% of the testbed has less than 13 pixels error distance for the probabilistic PBOD while the kernel and SIFT threshold for 50% accuracy are 24 and 54 pixels respectively.

Table 7. Comparison of the centroid distance among Probabilistic PBOD, kernel tracker and SIFT-based tracker.

Distance Error (Pixel)	Number of Image Pairs		
	Probabilistic PBOD	Kernel Tracker	SIFT-Based Tracker
0-9	52	45	40
10-19	30	34	19
20-29	19	24	9
30-39	13	6	3
40-49	3	2	6
50-59	0	5	8
60-69	1	1	2
70-79	0	0	0
80-89	0	0	2
90-99	0	0	2
>99	2	3	30

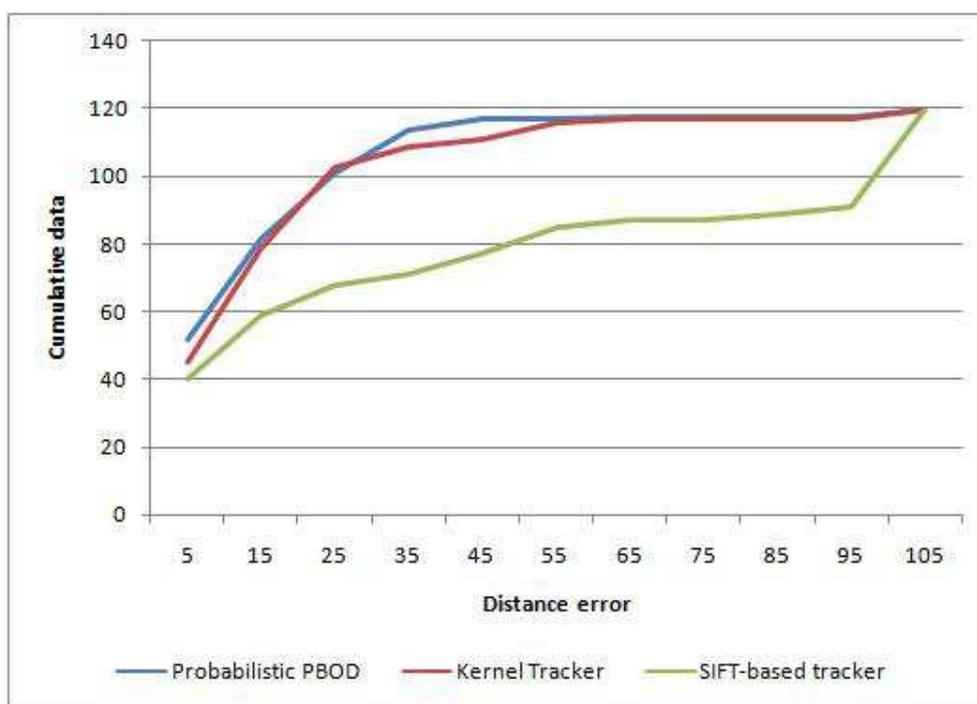
Table 8. Comparison of the average distance error among the PBOD, Kernel Tracker and SIFT-based tracker.

Methods	Probabilistic PBOD	Kernel Tracker	SIFT-Based Tracker
Average \mathcal{R}_D (%)	10.03	15.88	35.93

The main reason for our better performance compared with the method by Yan *et al.* is due to wider search area. Our method built the candidate patches at strategic locations throughout the whole frame, while the sub-templates are built on limited search space. If the object movement is fast, the search area must be wide enough, or else the candidate template or patch will not be generated. Another limitation for the method by Yan *et al.* is that the size variation is assumed to be constant in all directions. The size of the object may increase or decrease in one direction only such as for the case of a human extending his hand, where the size increment will be on that particular hand movement. Our method permits the size smoothing to be performed at certain direction only, which explains better precision. On the other hand, a tracker based on feature detection such as SIFT will not work well if the tracked object

has a non-rigid shape. SIFT's signature is built by collecting the neighbourhood data into a histogram of gradient where any background change will heavily affect its value. It also performs poorly if the image is blurred and has shadow noise since the keypoint's descriptor will be different and no match is found. This affects the detection and matching accuracy of the features, which will result in inaccurate observation. Another limitation of SIFT-based tracker is longer computational time, where it is heavily dependent on the number of points detected.

Figure 16. Cumulative distribution of error distance between probabilistic PBOD, Kernel tracker and SIFT-based tracker.



5. Conclusions

In this paper, we have shown that probabilistic PBOD works better compared with the deterministic approach in obtaining observation for single object tracking. Probabilistic PBOD registered 48.33% detection with less than 10 pixels error while the deterministic approach only achieved 42.50%. Both PBODs work well in challenging scenes, especially for the problems of low image sharpness, moderate deformation, illumination change, blur, size variation and homogeneous texture, by fusing feature- and template-based approaches. Probabilistic PBOD also performs better than kernel tracker by Yan *et al.* and SIFT-based tracker. The main novelties of probabilistic PBOD are (1) a probabilistic approach to patch-based object recognition, (2) modelling histogram matching by using Poisson and Gaussian distributions, and (3) statistically-based position and size smoothing for better detection accuracy. Robust observation detection allows the algorithm to improve track retention, especially during challenging scenes as the track can still obtain a measurement. This system can be further improved by implementing a stereo vision system as implemented by Marron-Romera *et al.* [38], which results in better detection and tracking accuracy as compared with a single camera system.

References

1. der Spek, S.V.; Schaick, J.V.; Bois, P.D.; Haan, R.D. Sensing human activity: GPS tracking. *Sensors* **2009**, *9*, 3033–3055.
2. Wren, C.; Azarbayejani, A.; Darrell, T.; Pentland, A. Pfindex: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 780–785.
3. Comaniciu, D.; Ramesh, V.; Meer, P. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **2003**, *25*, 564–577.
4. Zulkifley, M.A. Topics in Robust Video Analytics. PhD Thesis, The University of Melbourne: Victoria, Australia, 2012.
5. Zulkifley, M.A.; Moran, B.; Rawlinson, D. Robust foreground detection: A fusion of masked greyworld, probabilistic gradient information and extended conditional random field approach. *Sensors* **2012**, *12*, 5623–5649.
6. Zulkifley, M.A.; Moran, B. Robust hierarchical multiple hypothesis tracker for multiple-object tracking. *Expert Syst. Appl.* **2012**, *39*, 12319–12331.
7. Bradski, G.; Kaehler, A. *Learning OpenCV*; O'Reilly Media: Sebastopol, CA, USA, 2008.
8. Schiele, B.; Crowley, J.L. Object Recognition using Multi-dimensional Receptive Field Histograms and its Robustness to View Point Changes. In *Proceedings of the European Conference on Computer Vision*, Cambridge, UK, 15–18 April 1996.
9. Swain, M.J.; Ballard, D.H. Color indexing. *Int. J. Comput. Vis.* **1991**, *7*, 11–32.
10. Bhattacharyya, A. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calc. Math. Soc.* **1943**, *35*, 99–109.
11. Zulkifley, M.A.; Moran, B. Statistical Patch-Based Observation for Single Object Tracking. In *Image Analysis and Processing*; Maino, G., Foresti, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6979, pp. 119–129.
12. Chang, S.L.; Chen, L.S.; Chung, Y.C.; Chen, S.W. Automatic license plate recognition. *IEEE Trans. Intell. Trans. Syst.* **2004**, *5*, 42–53.
13. Khan, N.Y.; Imran, A.S.; Ali, N. Distance and Color Invariant Automatic License Plate Recognition System. In *Proceedings of the International Conference on Emerging Technologies*, Islamabad, Pakistan, 12–13 November 2007; pp. 232–237.
14. Syed, Y.A.; Sarfraz, M. Color Edge Enhancement Based Fuzzy Segmentation of License Plates. In *Proceedings of the Ninth International Conference on Information Visualisation*, London, United Kingdom, 6–8 July 2005; pp. 227–232.
15. Hsieh, J.W.; Peng, C.S.; Fan, K.C. Grid-based Template Matching for People Counting. In *Proceedings of the IEEE 9th Workshop on Multimedia Signal Processing 2007*, Crete, Greece, 1–3 October 2007; pp. 316–319.
16. Torralba, A.; Fergus, R.; Freeman, W.T. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 1958–1970.
17. Cole, L.; Austin, D.; Cole, L. Visual Object Recognition using Template Matching. In *Proceedings of the Australasian Conference on Robotics and Automation*, Canberra, Australia, 6–8 December 2004.

18. Yan, Y.; Huang, X.; Xu, W.; Shen, L. Robust kernel-based tracking with multiple subtemplates in vision guidance system. *Sensors* **2012**, *12*, 1990–2004.
19. Lowe, D.G. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Corfu, Greece, 20–25 September 1999; Volume 2, pp. 1150–1157.
20. Ke, Y.; Suthankar, R. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proceedings of the IEEE Computer Society Conference Computer Vision and Pattern Recognition*, Washington, United States of America, 27 June–2 July 2004; Volume 2, pp. 506–513.
21. Burghouts, G.J.; Geusebroek, J.M. Performance evaluation of local colour invariants. *Comput. Vision Image Underst.* **2008**, *113*, 48–62.
22. Geusebroek, J.M.; Smeulders, A. W. M.; Geerts, H. Color invariance. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 1338–1350.
23. Mikolajczyk, K.; Schmid, C. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1615–1630.
24. Ledwich, L.; Williams, S. Reduced Sift Features for Image Retrieval and Indoor Localisation. In *Proceedings of the Australian Conference on Robotics and Automation*, Canberra, Australia, 6–8 December 2004.
25. Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L.V. Speeded-Up Robust Features (SURF). *Comput. Vision Image Underst.* **2008**, *110*, 346–359.
26. Bradski, G.R. Computer vision face tracking for use in a perceptual user interface. *Intel Technol. J.* **1998**, *Q2*, 705–720.
27. Chen, Y.L.; Liang, W.Y.; Chiang, C.Y.; Hsieh, T.J.; Lee, D.C.; Yuan, S.M.; Chang, Y.L. Vision-based finger detection, tracking, and event identification techniques for multi-touch sensing and display systems. *Sensors* **2011**, *11*, 6868–6892.
28. Fukunaga, K.; Hostetler, L.D. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Inf. Theory* **1975**, *21*, 32–40.
29. Comaniciu, D. Bayesian Kernel Tracking. In *Proceedings of the Annual Conference of the German Society for Pattern Recognition*, Zurich, Switzerland, 16–18 September 2002; pp. 438–445.
30. Li, Z.; Xu, C.; Li, Y. Robust Object Tracking Using Mean Shift and Fast Motion Estimation. In *Proceedings of International Symposium on the Intelligent Signal Processing and Communication Systems, ISPACS 2007*, Chengdu, China, 6–8 December 2007; pp. 734–737.
31. Namboodiri, V.P.; Ghorawat, A.; Chaudhuri, S., Improved Kernel-Based Object Tracking Under Occluded Scenarios. In *Computer Vision, Graphics and Image Processing*; Kalra, P., Peleg, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4338, pp. 504–515.
32. Peng, N.S.; Yang, J.; Liu, Z. Mean shift blob tracking with kernel histogram filtering and hypothesis testing. *Pattern Recognit. Lett.* **2005**, *26*, 605–614.
33. Leichter, I.; Lindenbaum, M.; Rivlin, E. Mean shift tracking with multiple reference color histograms. *Comput. Vision Image Underst.* **2009**, *114*, 400–408.

34. Shi, J.; Tomasi, C. Good Features to Track. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 21–23 June 1994; pp. 593–600.
35. Weijer, V.D.J.; Gevers, T.; Smeulders, A.W.M. Robust photometric invariant features from the color tensor. *IEEE Trans. Image Process.* **2006**, *15*, 118–127.
36. Tsai, V.J.D. A comparative study on shadow compensation of color aerial images in invariant color models. *IEEE Trans. Geosci. Remote Sens.* **2006**, *44*, 1661–1671.
37. Kay, S.M. *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*; Prentice Hall: Upper Saddle River, NJ, USA, 1998.
38. Marron-Romera, M.; Garcia, J.C.; Sotelo, M.A.; Pizarro, D.; Mazo, M.; Canas, J.M.; Losada, C.; Marcos, A. Stereo vision tracking of multiple objects in complex indoor environments. *Sensors* **2010**, *10*, 8865–8887.

© 2012 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).