Louisiana State University
LSU Digital Commons

LSU Master's Theses | Graduate School

2011

Comparison of different integral histogram based tracking algorithms

Sriharsha Atluri
Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses

 Part of the Engineering Science and Materials Commons

Recommended Citation

Atluri, Sriharsha, "Comparison of different integral histogram based tracking algorithms" (2011). *LSU Master's Theses*. 1263.
https://digitalcommons.lsu.edu/gradschool_theses/1263

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.

COMPARISON OF DIFFERENT INTEGRAL HISTOGRAM BASED TRACKING ALGORITHMS

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Engineering Science

in
The Interdepartmental Program in Engineering Science

By
Sriharsha Atluri
B.Tech. Jawaharlal Nehru Technological University, 2008
May 2011

ACKNOWLEDGEMENTS

During the course of writing my thesis, many people have been supporting my work, and this thesis is a good opportunity to express my gratitude.

First of all, I would like to thank my parents for their constant support and unconditional love, without which this would not be possible. I even would like to thank my uncle and aunt for providing me with moral support and for having my back.

I would also like to thank my advisor, Dr. Bahadir K. Gunturk for introducing me to this interesting field of Image Processing. I would like to thank my advisor for giving me encouragement, courage and above all propelling me forward to complete this thesis. I thank my advisor for staying with me throughout my ups and downs. I thank Dr. Gerald M. Knapp, Graduate Advisor for Engineering Science program for allowing me to pick up my field of interest. I thank Dr. Suresh Rai for accepting my request to be in my Committee.

I shall thank my fellow researchers Robert Dibiano, Sertan Kaya for their support during the analysis, implementation & coding and testing phase of my project. I would like to thank them for giving their resources and time in enabling me to complete my thesis. Last but not the least; I would like to thank my roommates and friends with whom I spent my entire Master's program, for their constant support.

I would like to thank all those who were directly or indirectly involved with my work and whom I forgot to mention over here by name.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Object tracking is an important subject in computer vision with a wide range of applications – security and surveillance, motion-based recognition, driver assistance systems, and human-computer interaction. The proliferation of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated video analysis have generated a great deal of interest in object tracking algorithms. Tracking is usually performed in the context of high-level applications that require the location and/or shape of the object in every frame. Research is being conducted in the development of object tracking algorithms over decades and a number of approaches have been proposed. These approaches differ from each other in object representation, feature selection, and modeling the shape and appearance of the object.

Histogram-based tracking has been proved to be an efficient approach in many applications. Integral histogram is a novel method which allows the extraction of histograms of multiple rectangular regions in an image in a very efficient manner. A number of algorithms have used this function in their approaches in the recent years, which made an attempt to use the integral histogram in a more efficient manner. In this paper different algorithms which used this method as a part of their tracking function, are evaluated by comparing their tracking results and an effort is made to modify some of the algorithms for better performance. The sequences used for the tracking experiments are of gray scale (non-colored) and have significant shape and appearance variations for evaluating the performance of the algorithms. Extensive experimental results on these challenging sequences are presented, which demonstrate the tracking abilities of these algorithms.

CHAPTER 1: INTRODUCTION

Computer vision has become a major part of many day-to-day applications. Object tracking is an important aspect within the field of computer vision. Use of object tracking algorithms has been extensive in a wide range of domains including vehicle navigation, traffic monitoring, automated surveillance, video indexing, and human-computer interfaces. All these applications pose significant challenges to any object tracking algorithm when real time constraints are take into consideration such as abrupt object motion, changing appearance patterns of the object and its background, object occlusions, and noise.

A standard video has around 24 frames per sec and this number varies between different video formats or transmission schemes. When an attempt to implement an object tracker is made, the computational cost has to be as low as possible to meet the high frame rate of a video. A good algorithm is evaluated by the efficiency and accuracy at which it tracks the target and the time it takes to process each frame. Efficiency of a tracker can be considered as the overall performance of the tracker over a video sequence whereas the accuracy of a tracker is determined by how accurately the target is located by the tracker in successive frames.

Integral histogram [15] is an efficient algorithm which accelerates the extraction of histograms of multiple rectangular regions in an image plane. This technique enables to employ an exhaustive search all over the image plane in a relatively small time cost, thus enabling to track fast objects even in high frame rates that have significant relocation of the targets. A number of algorithms were proposed based on this method which utilized the luxury provided by the integral histogram function. Each of these algorithms has a different approach for feature selection of the target, and has its own advantage over one another when overall performance of the tracker is considered.

1.1 PROBLEM STATEMENT

Integral histogram technique [15] allows the computation of histograms of rectangular regions in an image using simple arithmetic operations. This function which when used efficiently helps to construct a good feature vector representing the target that can be used for the matching function. There are number of algorithms that have the integral histogram function as an important part of their tracking algorithms. Most of these algorithms have their own approach of constructing a good feature vector to better represent the target, so that the tracker will be able to track it under extreme conditions. Some of the issues that were encountered in these algorithms are:

- Using a single window may not capture the local characteristics of the target.
- Failure to construct a good feature vector from the target's features, which will be used in the matching function of the algorithm.
- The lack of a robust similarity criterion that uses both color and spatial properties of the target.
- Failure to include the possibility of target occlusions and shape and appearance variations of the target, during the tracking process.
- Use of the target template as a global model which may not handle significant changes in the target very well.

1.2 OBJECTIVES

Many histogram-based tracking algorithms require extraction of intensity or color histograms over a large number of sub-windows in the target image and in the object template. Integral histogram data structure [18] has become a key tool, by making these histogram extractions possible using simple arithmetic operations, enabling the application of many algorithms for real time tracking tasks. These algorithms made an attempt to exploit this technique in their approaches as it reduces the computational cost by a considerable amount. These algorithms used different approaches in including the integral histogram function in their tracking functions.

The objectives of this research were to:

- Investigate the possibility of an effective and robust real time tracking function based on the integral histogram technique.
- Improve the robustness of the tracking function against partial or full occlusions.
- Improve the usage of the spatial information in computing the model feature vector of the target for good performance.
- Evaluate the performances of integral histogram based tracking algorithms.

CHAPTER 2: LITERATURE REVIEW

2.1 OBJECT TRACKING

Object tracking is an important subject in computer vision with a wide range of applications including security and surveillance, driver assistance systems, traffic monitoring, and human-computer interfaces. What makes tracking difficult is the potential variability of a target in a video, with respect to shape and appearance. This variability arises from variations in target pose and partial or full occlusion of the target as the image sequence progresses. When not taken into account, any one of these variations is enough to cause a tracking algorithm to lose its target during tracking. So, for any tracking algorithm, the performance is judged by how accurately it can track an object and the computational cost needed for the tracking.

In order to track objects, certain features of the objects have to be selected which should be identifiable under varying poses and over a large number of frames. Many tracking algorithms were proposed to overcome the obstacles that arise from noise, partial or full occlusions, and shape variations. Intensity histograms are popular representations of object of interest, but they disregard the spatial arrangement of the feature values. Efforts have been made to include the spatial properties [16] [18] in addition to the color histograms in constructing a robust similarity criterion that could be effective enough against the common obstacles.

Figure 1: Examples of object tracking [16].

2.2 FEATURE-BASED TRACKING ALGORITHMS

Feature-based tracking algorithms are used frequently in applications where time factor is critical. Feature-based tracking is a reliable approach, in which moving objects are represented by a set of feature points, clustering them into higher level features and then matching the features between images. These features may be defined by the user prior to the tracking or can be extracted during the tracking [1]. When in motion, all points in the target region are presumed to be part of the same object, which in-turn allows the assumption that these points move coherently in space. So, all the feature-based tracking algorithms assume a structural form to model an object's motion. Traditionally, motion has been mostly represented as either translational or affine model [2], which indeed has proved to be reliable for small, linear movements. In order to achieve stability and robustness against occlusions, filters have been used to smooth the object trajectory.

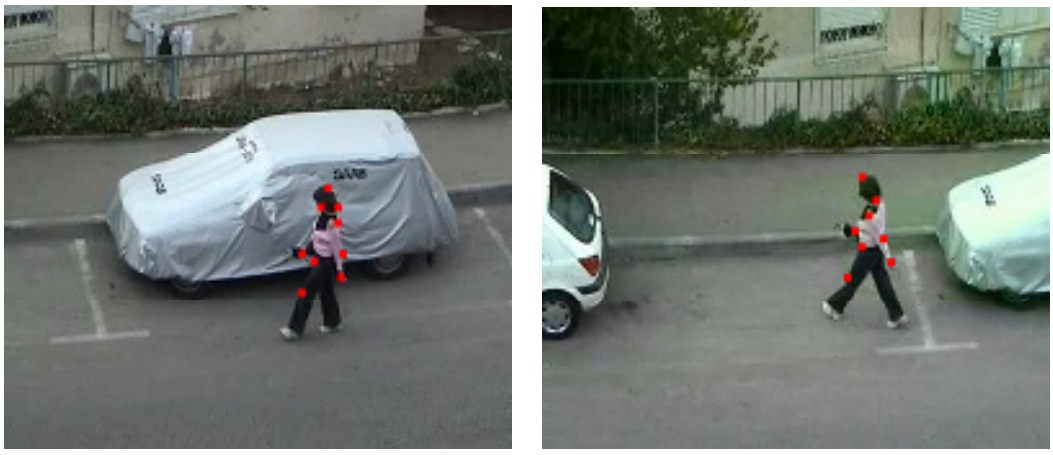


Figure 2: An example of feature-based tracking [3].

In these algorithms, several feature point selection strategies are used. The goal is to obtain distinctive feature points on the image that are appropriate for tracking. However, when tracking over a large image sequence, the structural models become complex as geometric deformations of the target become more and more significant and it becomes difficult for a

translational or affine model to be assumed as the object's trajectory. Another challenging problem for feature-based object tracking is that ambiguity often arises when a feature point in one frame has many similar points in another frame [3]. To suppress ambiguity, algorithms often perform an exhaustive search over large windows and as a result the computational complexity of the algorithms is increased considerably. The present feature-based tracking algorithms are very time efficient but less accurate and extensive research is being conducted over the last decade to improve the performance criteria. Though a number of modifications were done by imposing several constraints on the tracker, a less number of attempts were made to improve the tracking by selection of good feature points.

2.3 CONTOUR-BASED TRACKING ALGORITHMS

Tracking deforming objects involves estimating the global motion of the object and its local deformations as a function of time. Tracking algorithms have been using block or particle filters for estimating finite dimensional representations of shape, but these algorithms are dependent on a number of parameters and cannot handle changes in curve topology efficiently. Geometric contours [6] provide a platform which is parameter independent and allow for changes in topology. Contour-based tracking algorithms track targets by initially extracting and representing the target's outline as bounding contours. Active geometric contours algorithms add another step to the process by updating the contours dynamically in successive frames. These algorithms are highly dependent on the initialization of the tracking, which makes the tracking difficult if it starts automatically [7]. The initialization process has to be done manually for better performance.

Figure 3: An example of contour-based tracking [5].

If the motion of the target or region of interest is simple, only a small number of variables are required to represent the contour motion. However, if the target is changing rapidly between frames, each contour point can move independently [5]. Contour deformation then forms an infinite dimensional space. Direct application of particle filters for large dimensions is impractical, because of the reduction in particle size as dimension increases. A major drawback is that most of these methods do not attempt to include any shape information of the object [4], which may be vital for tracking objects that are expected to go considerable changes. And as stated above for active geometric contours, the initialization of the tracking is important and these methods may perform poorly when the recognized target has partial or full occlusions, as the contours are updated for every frame.

2.4 REGION-BASED TRACKING ALGORITHMS

When compared to the above two classifications, the region-based algorithms are the most frequently used algorithms. In the region-based category, modeling of the target by a histogram or by other non-parametric features has become very popular in recent years. For these algorithms, the user can take into account both the spatial and intensity information of the object. In this category of algorithms, there are number of parameters that can be used to track an object

depending on the application requirements. The region-based algorithms are preferential over the other algorithms when performance is taken into consideration.

One such approach in the region-based tracking algorithms is the mean-shift tracker. In this method, the tracker tries to find the image window that is most similar to the object's color histogram in the current frame. It iteratively carries out a kernel based search starting at the previous location of the object [9]. Even though there are variants to improve its performance by performing additional operations, the original method requires the target kernels in the successive frames to have a minimum overlap. The success of the mean-shift highly depends on the discriminating power of the histograms. Though simple and efficient to compute, the mean-shift tracker uses a color histogram which only describes the color distribution and ignores spatial information or layout of the colors [8]. This inadequacy would often cause problems especially when similar color distributions exist in the target surroundings. However, mean-shift owns its speed to the fact that it only evaluates the similarity within a limited search region.

Kernel-based tracking methods [10] have recently gained popularity due to their range of convergence and their robustness to object deformations. One of the most appealing merits of kernel-based trackers is their low computational cost, compared with other commonly employed tracking schemes, such as particle filters or exhaustive template matching. In addition, multiple kernels help increase the measurement space, sensitivity and in general, the structure of the kernel-based tracking. Various enhancements to handle particular problems arising from these algorithms such as scale selection, feature fusion, etc., which are critical features for a tracking procedure, have yet to be included in the kernel based algorithms. Convergence properties [11] of the mean-shift are improved by integrating background and template similarities in the iterative update mechanism.

2.5 ROLE OF KALMAN FILTERS

A common approach is to employ predictive filtering and use the statistics of object's color and location in the distance computation. One such filter that is being used extensively in object tracking is Kalman filters [12]. Kalman filter works based on the assumption that all distributions are Gaussian. A Kalman filter is used to estimate the state of a linear system where the state is assumed to be distributed by a Gaussian. In practice, for many linear or linearizable systems this assumption often works reasonably as far as the noise in the system dynamics is concerned. Kalman filter can be used as a probabilistic prediction technique to make tracking more robust.

Uni-modal Kalman filtering offers the advantage of a closed form temporal fusion and its limitations can be dealt with by choosing features more carefully. Given the complications arising from the infinite dimensional nature of the space of smooth, many approaches propose estimator designs requiring manual gain selection. In some algorithms, Kalman filter in conjunction with active contours to track non-rigid objects are used [13]. The Kalman filter is used for predicting possible movements of the object, while the active contours allowed for tracking deformations in the object.

Particle filtering provides fusion of different sensor data, to incorporate constraints and to account for different uncertainties. A particle filter is used to represent the tracking system's state, and a method of accelerating the likelihood calculation of the filter is developed. Measurements for each particle are not independent, Posterior and prior distributions [14] are no longer limited to single Gaussians but can adopt truly non-Gaussian, multi-modal forms. However, particle filters are flexible in terms of motion models supported; this allows the correct integration of a fast motion estimation algorithm which produces very noisy rotation estimates.

2.6 TEMPLATE UPDATE vs FIXED TEMPLATE TRACKING

For most of the algorithms, when considering the transition from frame to frame, there are two approaches which are template update and non-template update. These approaches are used in the algorithms based upon the requirements of the application and the features that are being used for the tracking algorithm. In non-template update, the initial target template is used as global model for target location over a large number of frames. In this approach, certain features of the objects have to be selected which should be identifiable under varying poses. For non-template update approach, considering the target as a single block does not do much good for the tracking process as it does not contain the spatial properties and color distribution information of the target. Efforts [16] [9] were made to include the spatial and intensity or color properties of the target in an efficient manner to compute a robust similarity criterion for better performance against pose variations and occlusions.

In a template update approach, the target template is updated to its new recognized target after its location in every frame. This is a more robust approach for a tracking algorithm but at the expense of an increase in computational cost. These target template updates cannot be simple updates to cope with the common tracking problems. As in [17], different blocks are formed around prominent features of the target with minimum overlap and weights are given to these blocks depending on their foreground and background pixel information. These weights are updated in each frame after target location and updating, as there can be significant changes in the features of the target. As stated earlier, these methods are practiced based upon the algorithms approach for the common tracking problems.

CHAPTER 3: INTEGRAL HISTOGRAM BASED ALGORITHMS

In integral histogram function [15], the spatial arrangement of data points is exploited by recursively propagating an aggregated function, starting from the top-left corner of an image and traversing through the remaining points in a string scan fashion i.e., left to right and top to bottom in an image. At each step, a single bin is updated using the values of integral histogram of the previously calculated neighboring data points. After the histogram function is propagated through the entire image, histogram of any rectangular region can be calculated easily by simple arithmetic operations.

Let $h(x, y, n)$ be the histogram of an image from $(1, 1)$ to (x, y) and n is the index of the histogram bin, $1 \leq n \leq N$, where N is the total number of bins. $h(x, y, n)$ is called the integral histogram. The integral histogram calculation algorithm is given in Algorithm 1.

Algorithm 1: Integral histogram calculation

Set the number of bins, N

Set the histograms for out of bound pixels to zero $h(x, y, n) = 0$ *for* $x \leq 0$ or $y \leq 0$

For each pixel location (x, y) **in the image**

Get the neighboring histograms $h(x-1, y, n)$, $h(x, y-1, n)$, $h(x-1, y-1, n)$

Get the current pixel value $f(x, y)$

Compute the quantized bin value of the current pixel

$$Q(f(x,y),n) = \begin{cases} 1 \; if \; f(x,y) \; falls \; into \; n \; th \; bin \\ 0 \; otherwise \end{cases}$$

Compute the current pixel histogram $h(x, y, n)$

$$h(x, y, n) = h(x-1, y, n) + h(x, y-1, n) - h(x-1, y-1, n) + Q(f(x, y), n)$$

End

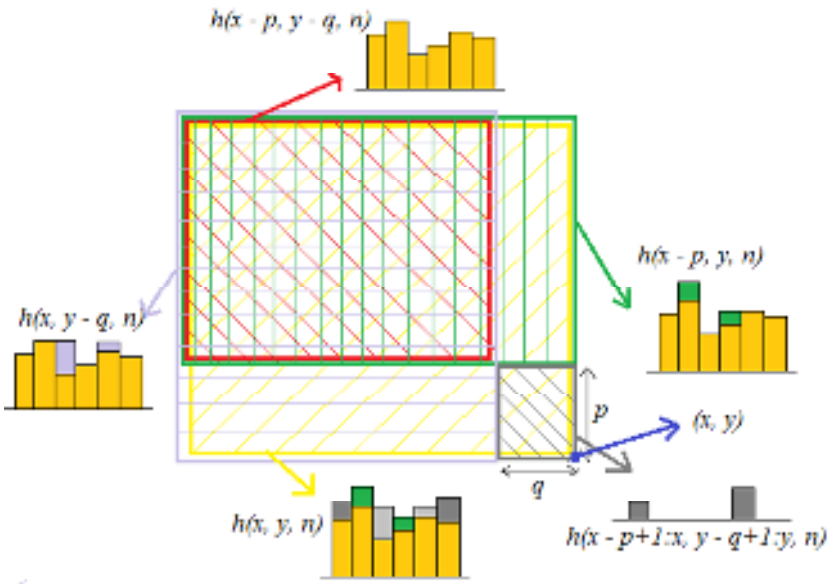


Figure 4: Histogram extraction [9].

Once the integral histogram is calculated, a local histogram is computed very easily from it. An illustration of the local histogram calculation is shown in Figure 4. Lets say the block of interest is the grey block and its histogram has to be computed. (x, y) are the coordinates of the right-bottom corner pixel of the block. The block has a height of p and width of q.

$\hat{h}(x, y, n) = h(x - p + 1{:}\, x, y - q + 1{:}\, y, n)$, where $\hat{h}(x, y, n)$ denotes the local histogram of the block formed at x and y. The histogram of the block is given by:

$$\hat{h}(x, y, n) = h(x, y, n) - h(x - p, y, n) - h(x, y - q, n) + h(x - p, y - q, n) \quad \text{..... (1)}$$

There are a number of advantages using the Integral histogram as a part of the tracking function. It is computationally superior to the usual approach because of the simple procedure for histogram extraction of region of interest. This property helps in employing an exhaustive search all over the image plane in a relatively small time cost, thus enabling to track objects even in high frame rates that have significant relocation. When using the integral histogram, histogram extraction of a region does not depend on size of the target of interest. Thus, integral histogram makes the histogram extraction process independent of size of the target.

3.1 SINGLE BLOCK BASED MATCHING

The target is represented by using a single block, typically initialized at the first frame with reference coordinates (x_o, y_o) and its histogram is used as the feature vector for the matching function, in order to observe the performance of the tracker and try to recognize the defects of using single block based matching.

Two distance/similarity measures are taken into consideration for the matching function, namely Euclidean distance and Bhattacharya distance.

Euclidean distance map is a dissimilarity measure given by:

$$\rho(x,y) = \frac{1}{N}\sum_{n=1}^{N}(\hat{h}(x_o, y_o, n) - \hat{g}(x,y,n))^2 \quad \text{..... (2)}$$

Bhattacharya distance is a similarity measure given by:

$$\rho(x,y) = \frac{1}{N}\sum_{n=1}^{N}\hat{h}(x_o, y_o, n)\hat{g}(x,y,n) \quad \text{..... (3)}$$

where *N* represents the number of bins, $\hat{h}(x_o, y_o, n)$ represents the local histogram of the target initialized at the first frame, $\hat{g}(x,y,n)$ represents the local histogram of the block at (x,y) coordinates.

The local histogram of the block $\hat{g}(x,y,n)$ formed at pixel location (x,y) is computed from the *g(x, y, n)* which is the integral histogram of the frame in which the target has to be located. The single block based matching algorithm is given in Algorithm 2.

Algorithm 2: Single block based matching algorithm

Set the number of bins, *N*

Initialize the target and get its histogram $\hat{h}(x_o, y_o, n)$

For each frame

Compute the integral histogram of the frame $g(x,y,n)$

Derive the local histogram $\hat{g}(x,y,n)$ **for every point** (x,y)

Calculate the distance map $\rho(x, y)$ at every point (x, y)

Determine the estimated target position $(\hat{x}, \hat{y}) = argmin_{(x,y)} \rho(x, y)$

where $\rho(x, y)$ is the Euclidean distance map as in (2)

Update the target (for the template update approach)

$\hat{h}(x_o, y_o, n) = \hat{g}(\hat{x}, \hat{y}, n)$

End

3.2 SUB-BLOCK BASED MATCHING

In [16], it is proposed to improve the integral histogram by using multiple patches shown in Figure 5, to represent the target. The original template is represented by multiple blocks or patches and these patches are arbitrary and are not based on the target features. Every patch votes on the possible positions of the object in the current frame, by comparing its histogram with the corresponding image patch histogram. To get a similarity measure between the current frame and the target, the vote maps of the multiple patches are averaged as given by:

$$\rho(x, y) = \frac{1}{P}\sum_{i=1}^{P} \frac{1}{N} \sum_{n=1}^{N} \left(\hat{h}_i(x_o, y_o, n) - \hat{g}_i(x, y, n)\right)^2 \quad \text{..... (4)}$$

where *P* represents the number of blocks the target is divided into, *N* represents the number of bins, $\hat{h}_i$ represents the histogram of the *i* th block of the target, $\hat{g}_i$ represents the histogram of the *i* th block at *(x, y)* of input frame.

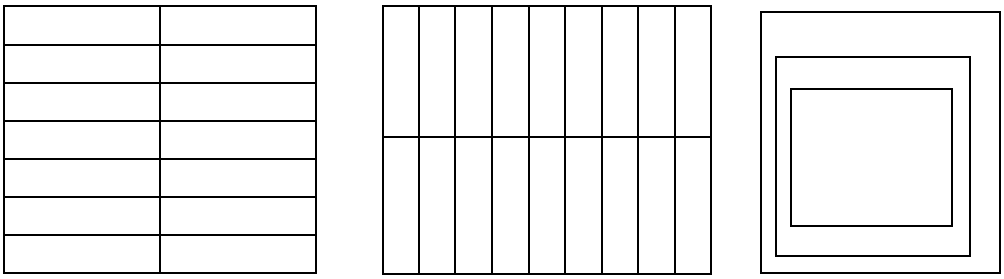


Figure 5: Different patches used [10].

This improvement to the integral histogram includes local spatial information and color or intensity distributions of the target in the construction of the feature vector, which may be vital for the tracking process. The robust nature of this algorithm allows it to use the algorithm without giving consideration to the number of blocks/ patches the target can be divided into. This algorithm gives the luxury of applying any similarity measure, which proves its efficiency and robustness.

The proposed algorithm is implemented by dividing the target into a set of blocks with equal size. The division of the target is practiced by dividing it into 3x3 blocks and 4x4 blocks as shown in Figure 6. A key feature matrix for the matching function is derived from these blocks, the number of columns of the matrix representing the number of blocks the target is divided into and each column representing the histogram of corresponding block of the target.

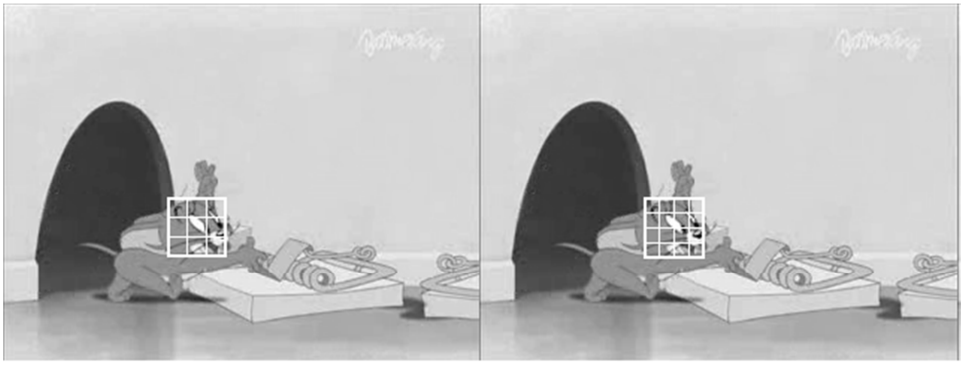


Figure 6: Different representations used in sub-block method.

Once the integral histogram of a frame is calculated, at each and every pixel a similar feature matrix is computed, by dividing the every region into the same number of blocks. The histogram extraction of each block did not take much time as the histogram extraction could be done using simple arithmetic operations because of the integral histogram. Updating the template at each frame is also practiced for this method. The algorithm for this approach is given in Algorithm 3.

Algorithm 3: Sub-block based matching algorithm

Set the number of bins, N

Declare the number of sub-blocks P

Compute the integral histogram of the target

Calculate the local histograms $\hat{h}_i(x_o, y_o, n),\ i = 1,2, \dots, P$

For each frame

Compute the integral histogram of the input frame

Construct a similar feature matrix for each pixel $\hat{g}_i(x, y, n),\ i = 1,2, \dots, P$

Calculate the distance map $\rho(x, y)$**at every point** (x, y) **as in (4)**

Determine the estimated target position $(\hat{x}, \hat{y}) = argmin_{(x,y)}\, \rho(x, y)$

Update the target (for the template update approach)

$$\hat{h}(x_o, y_o, n) = \hat{g}(\hat{x}, \hat{y}, n)$$

End

3.3 COVARIANCE TRACKING

In [17], an elegant algorithm is proposed using a covariance based object description and a Lie algebra based update mechanism. The object window is represented using a covariance matrix of features, managing to capture both the spatial and statistical properties as well as their correlation within the same representation. The update mechanism effectively adapts to the undergoing object deformations and appearance changes. At each frame, feature vectors of the object region are constructed using different spatial and intensity features like mean, variance, filter responses, intensity histograms. Then a covariance matrix is computed from these feature vectors and this becomes the feature model for the next frame. In the next frame, an area which has minimum covariance distance from this model is computed and assigned as the new estimated location of the object.

The use of covariance matrix can solve most of the traditional tracking problems. In general, a single covariance matrix extracted from a region is enough to match the region in different shapes and poses. Another advantage of the covariance matrix is that it is independent of the size of the target and varying illumination conditions. The time cost of this algorithm is entirely dependent on the number of feature vectors that are used for the computation of the covariance matrix. Though the covariance matrix is independent of the size of the object, its computation is dependent on the number of feature vectors used. As the number of feature vectors used to describe an object increase, so does its computation time.

For an M x N rectangular region, covariance matrix of the vectors $\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$……..$\mathbf{f}_k$

$$C_R = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{f}_i - \boldsymbol{\mu}_R)(\mathbf{f}_i - \boldsymbol{\mu}_R)^T \qquad (5)$$

where C_R is the covariance matrix, k indicates the number of feature vectors, T indicates a transpose function, $\boldsymbol{\mu}_R$ is the mean of the feature vectors given by $\boldsymbol{\mu}_R = \frac{1}{k}\sum_{i=1}^{k} f_i$.

Since the similarity measure used here has to measure the distance between two covariance matrices, Forstner distance [19] is used in the paper given by:

$$\rho(x, y) = \sqrt{\sum_{k=1}^{d} ln^2 \lambda_k (C_R, C(x, y))} \qquad (6)$$

where $\lambda_k (C_R, C)$ are the generalized Eigen values of C_R and C, d indicates the number of diagonal elements, C_R is the covariance matrix of the target, $C(x, y)$ is the covariance matrix computed at location x, y.

Instead of using derived features as proposed [17], the histograms of blocks are used to compute the covariance matrix of the target. To better describe the target and keeping the time cost in mind, the target is divided into 3x3 blocks. The histogram of each individual is calculated

which gives 9 feature vectors and then using these feature vectors, a covariance matrix is derived. The basic structure of the algorithm is given in Algorithm 4.

For the target divided into a 3x3 block, 9 feature vectors *(***$\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{h}_3$,….., $\mathbf{h}_9$***)* are derived from the 9 blocks formed. Covariance matrix of the regions is given by:

$$H_o = \frac{1}{9}\sum_{i=1}^{9}(\mathbf{h}_i - \boldsymbol{\mu}_R)(\mathbf{h}_i - \boldsymbol{\mu}_R)^T \quad \text{..... (7)}$$

where H_R is the covariance matrix, $\mathbf{h}_i$ is the histogram of the i[th] block, $\boldsymbol{\mu}_R$ is the mean of the feature vectors given by $\boldsymbol{\mu}_R = \frac{1}{9}\sum_{i=1}^{9} h_i$.

The similarity measure will be

$$\rho(x,y) = \sqrt{\sum_{k=1}^{d} ln^2\lambda_k(H_o, G(x,y))} \quad \text{..... (8)}$$

Algorithm 4: Covariance tracking algorithm

Set the number of bins, N

Compute the integral histogram of the target

Calculate the local histograms $\hat{h}_i(x_o, y_o, n),\ i = 1, 2, \dots, 9$

Compute the covariance matrix of the target H_o **as in (7)**

For each frame

Compute the integral histogram of the input frame

Construct a similar feature matrix for each pixel $\hat{g}_i(x, y, n),\ i = 1,2, \dots,9$

Compute the covariance matrix $G(x, y)$ **at every point** (x, y)

Calculate the distance map $\rho(x, y)$**at every point** (x, y) **as in (8)**

Determine the estimated target position $(\hat{x}, \hat{y}) = argmin_{(x,y)}\, \rho(x, y)$

End

3.4 ARTICULATING BLOCKS APPROACH

In [18], an accurate tracker is proposed using updated model of shape and appearance of the target, as the tracker progresses in the video. In this algorithm, the constantly changing foreground shape is modeled as a small number of rectangular blocks, whose positions within the tracking window are adaptively determined. An efficient representation of the target using histograms is proposed, so that it can be easily evaluated and compared. Shape update, which typically requires more elaborated algorithms, is carried out by adjusting a few small blocks within tracking window as shown in Figure 7. The irregular shape is approximated with a relatively small number of blocks that cover the foreground object with minimal overlaps. Once the target is recognized, updating the target shape by adjusting these blocks locally is done at each and every frame, so that they provide a maximal coverage of the foreground target as the sequence progresses. The proposed algorithm first locates the target by scanning the entire image using the estimated foreground intensity distribution. The refinement step that follows provides an estimated target contour from which the blocks can be repositioned and weighted.

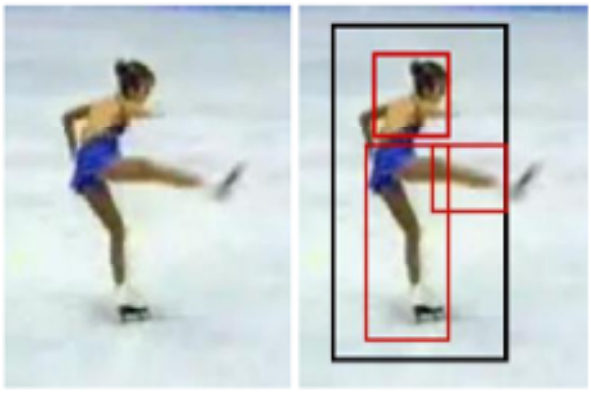
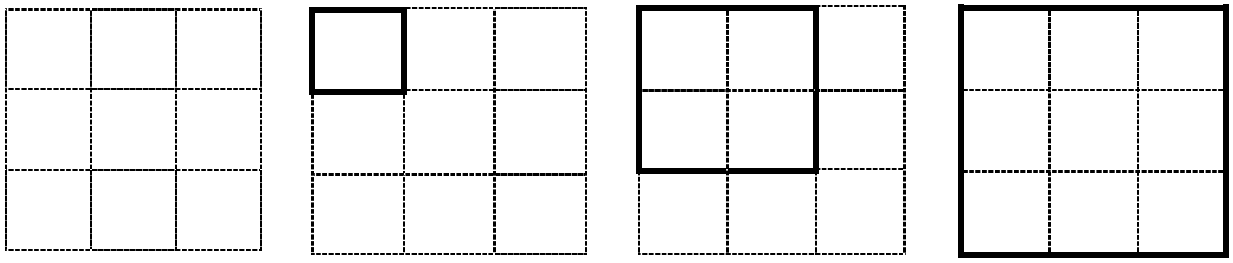


Figure 7: Tracking using articulating blocks [11].

3.5 MULTIPLE BLOCK BASED APPROACH

After going through the above paper, some modifications are picked from it. In an earlier method, the same approach of dividing the target into multiple blocks is used. In this approach, though the target is divided into a standard 3x3 sub block, all the block formations as shown in Figure 8, are used to get 14 feature vectors (i.e., 9 single blocks, 4 – 2x2 blocks, 1 – 3x3 block). A matrix is derived from these feature vectors of 14 columns, each column representing a block's histogram.



3x3 sub-blocks **9 single blocks** **4 2x2 blocks** **1 3x3 block**

Figure 8: 14 Blocks formed from a 3x3 block.

During the matching process, a similar matrix to the feature vector matrix is deduced at every pixel. Then in both the matrices, the distances between the corresponding columns are calculated separately. The final similarity measure is computed from the average of these individual distances as in equation (4), where the number of blocks is constant. This process is repeated for each and every pixel of the frame. The similarity measure used here is the Euclidean distance. Even though the algorithm looks a little complex and time consuming, the actual time take for each frame is low because of the effective usage of the integral histogram. The approach that is followed for this approach is given by Algorithm 5.

Algorithm 5: Multiple block based algorithm

Set the number of bins, N

Compute the integral histogram of the target

Calculate the local histograms $\hat{h}_i(x_o, y_o, n),\ i = 1,2,\dots,14$

For each frame

Compute the integral histogram of the input frame

Construct a similar feature matrix for each pixel $\hat{g}_i(x, y, n),\ i = 1,2,\dots,14$

Calculate the distance map $\rho(x, y)$**at every point** (x, y) **as in (4)**

Determine the estimated target position $(\hat{x}, \hat{y}) = argmin_{(x,y)}\, \rho(x, y)$

Update the target (for the template update approach)

$\hat{h}(x_o, y_o, n) = \hat{g}(\hat{x}, \hat{y}, n)$

End

3.5.1 DIFFERENT WEIGHTS

Initially, this algorithm is practiced by giving equal weights to all the blocks independent of their size. Then to improvise the result, weights are given to the blocks depending on their size. In one case, the bigger blocks are given more weight when compared to the smaller blocks. In another situation, it is vice versa i.e., the bigger blocks are given lesser weights when compared to the smaller blocks. A slight modification is made to equation (4) by adding weights (w_i) to the blocks.

$$(\hat{x}, \hat{y}) = argmin_{(x,y)} \frac{1}{\sum w_i} \sum_{i=1}^{14} w_i \frac{1}{N} \sum_{n=1}^{N} (\hat{h}_i(x_o, y_o, n) - \hat{g}_i(x, y, n))^2 \quad \text{..... (9)}$$

where w_i are the weights given to the blocks.

3.5.2 GIVING A THRESHOLD TO THE DISTANCE

During the matching process, when one of the blocks is compared to a completely different block, automatically the similarity measure between the blocks is going to be a high value. As the final similarity measure is taken as an average of the individual block distances, even one bad distance could affect the final measure. So as a modification to the above method, an effort is made to eliminate such bad measures by giving a threshold to the distances. When given a threshold, these bad similarity measures will have a less effect on the final similarity measure.

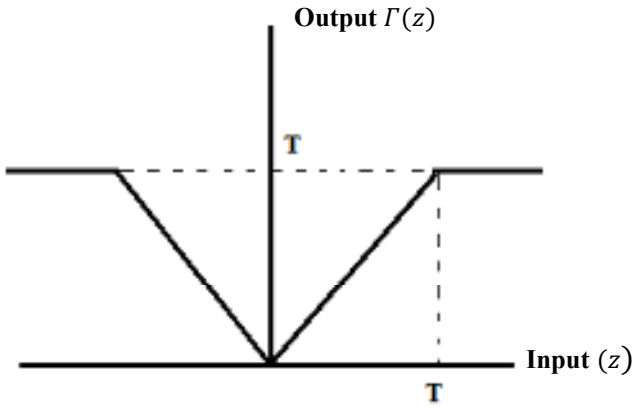


Figure 9: Illustrating the use of a threshold.

$$\hat{x},\hat{y} = \underset{x,y}{argmin}\frac{1}{14}\sum_{i=1}^{14}\Gamma\left(\frac{1}{N}\sum_{n=1}^{N}\left(\hat{h}_i(x_o,y_o,n) - \hat{g}_i(x,y,n)\right)^2\right) \quad \text{..... (10)}$$

where $\Gamma(z) = \begin{cases} |z| \ \ when\ |z| \leq T \\ T \ \ when\ |z| > T \end{cases}$

3.6 MOVING AVERAGE

During the tracking process, sometimes the trackers pick up objects which are similar to the target but not the actual ones due to the same intensity histograms. Sometimes due to shape and pose variations, the tracker will not be able keep up with the target and eventually lose them. To deal with these problems an algorithm is implemented, which is a slight modification to the previous approach.

Instead of comparing the present frame with the target recognized in the previous frame, an average of the last *M* recognized targets is used, to compensate with sudden changes in the object recognition. In this way, even though the tracker lost the target in one frame, due to the averaging technique it will be able to recognize the target in the next frame which ultimately helps in increasing the performance of the tracker. The averaging is initially started by giving considering the last 5 recognized targets, and moved it up to 10. The basic structure of the algorithm is given by Algorithm 6.

Algorithm 6: Moving average algorithm

Set the number of bins, N **and set the parameter** M

Compute the integral histogram of the target

Calculate the local histograms $\hat{h}_i(x_o, y_o, n),\ i = 1,2, \dots ,14$

For each frame p

- **Compute the integral histogram of the input frame**
- **Construct a similar feature matrix for each pixel** $\hat{g}_i(x, y, n),\ i = 1,2, \dots ,14$
- **Calculate the distance map** $\rho(x, y)$**at every point** (x, y) **as in (4)**
- **Determine the estimated target position** $(\hat{x}, \hat{y}) = argmin_{(x,y)}\, \rho(x, y)$
- **Update the target feature matrix** $\hat{h}_i(x_o, y_o, n) = \frac{1}{M} \sum_{m=0}^{M-1} \hat{g}_{p-m}(\hat{x}, \hat{y}, n)$

End

CHAPTER 4: RESULTS AND ANALYSIS

4.1 SINGLE BLOCK BASED MATCHING

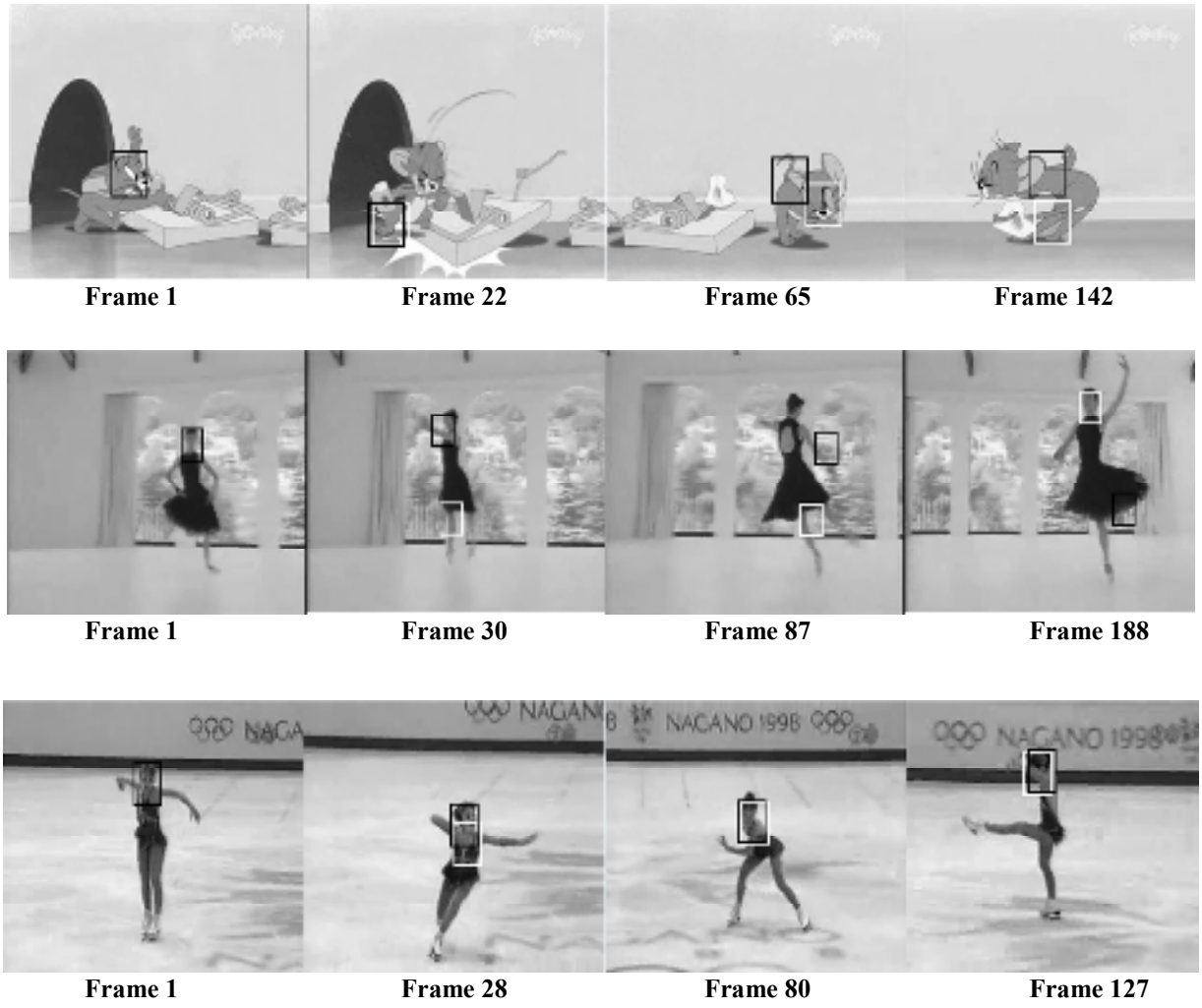


Figure 10: Results for single block based matching using Euclidean distance, the white block indicating the use of initial template and the black box indicating the use of a template update method.

From the results it can be concluded that when using a single block based matching function the performance of both the non-template update tracking and template update tracking approaches is very low. After noticing these results, a statement can be made that considering the target as a single block and using its histogram for identifying the target in the next frames, would not be sufficient for efficient and robust tracking.

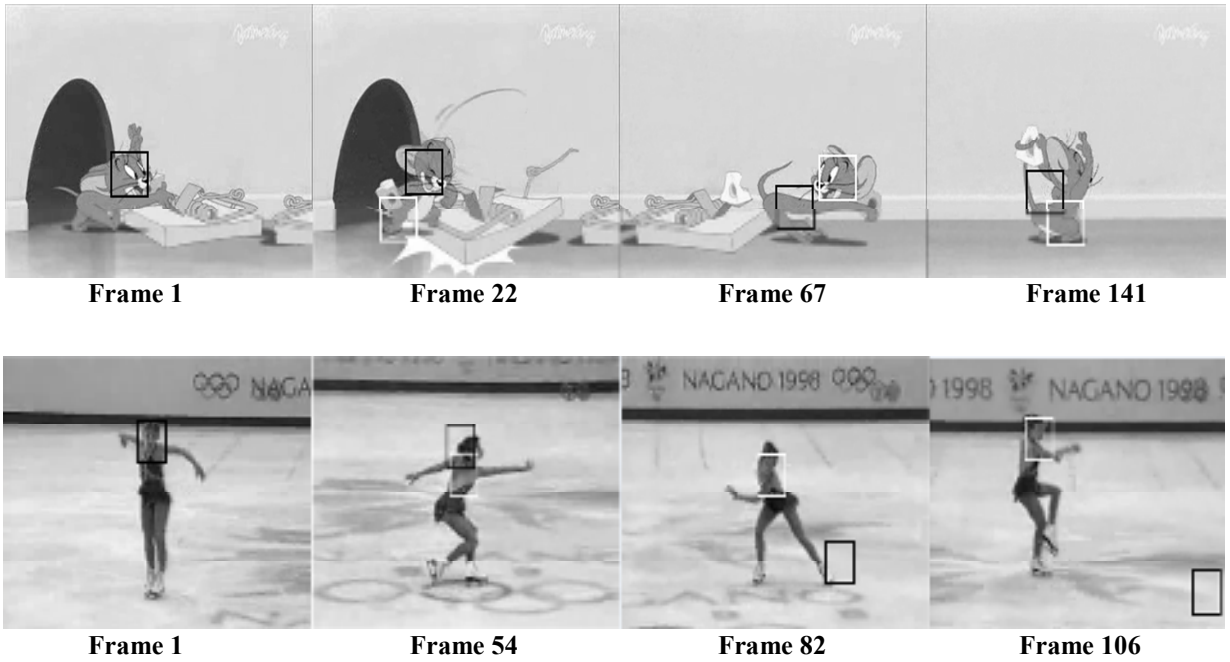


Figure 11: Results for single block based matching using Bhattacharya distance, the white block indicating the use of initial template and the black box indicating the use of a template update method.

The template update algorithm did not give results as the single block based matching function is not efficient and robust. A template update can only be used, when the original tracking algorithm is efficient and accurate so that the tracker does not lose the target in any case. If the target is lost when using an updated template, there is no way recovering the tracking process.

In this algorithm, the object of interest is used as a single block, which implies that the foreground and background pixels are given equal weights and cannot differentiate between the pixels that belong to the target and pixels that do not come in the target. So even though, the algorithm is simple and quick, it may be not efficient when employed for tracking purposes where the target may undergo shape and appearance variations. Another major drawback when the target is considered as a single block is loss of spatial information which is vital for tracking the object accurately.

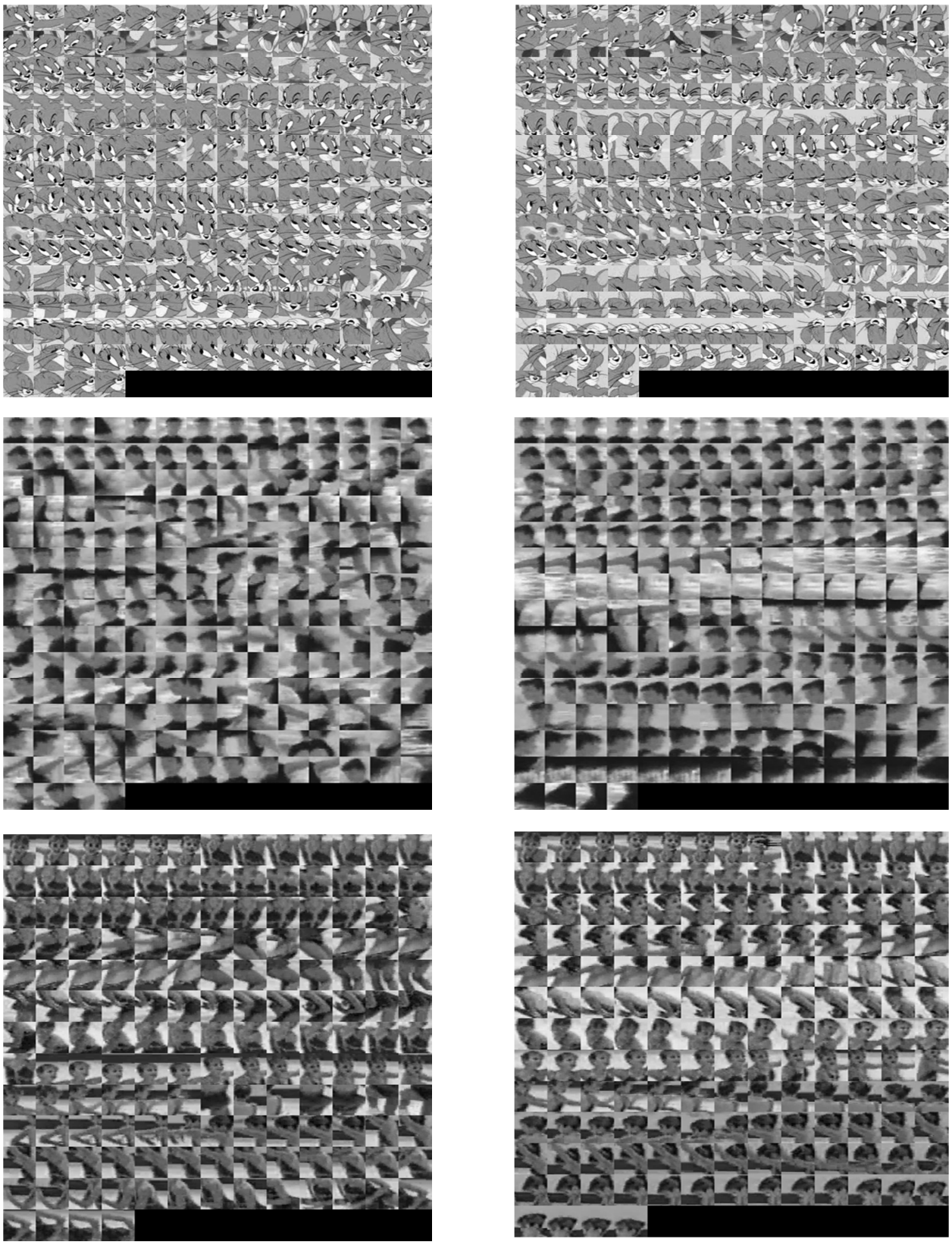


Figure 12: Performance of single block based matching using Euclidean distance, the left column indicates the use of initial template and the right column indicates the use of a template update method.

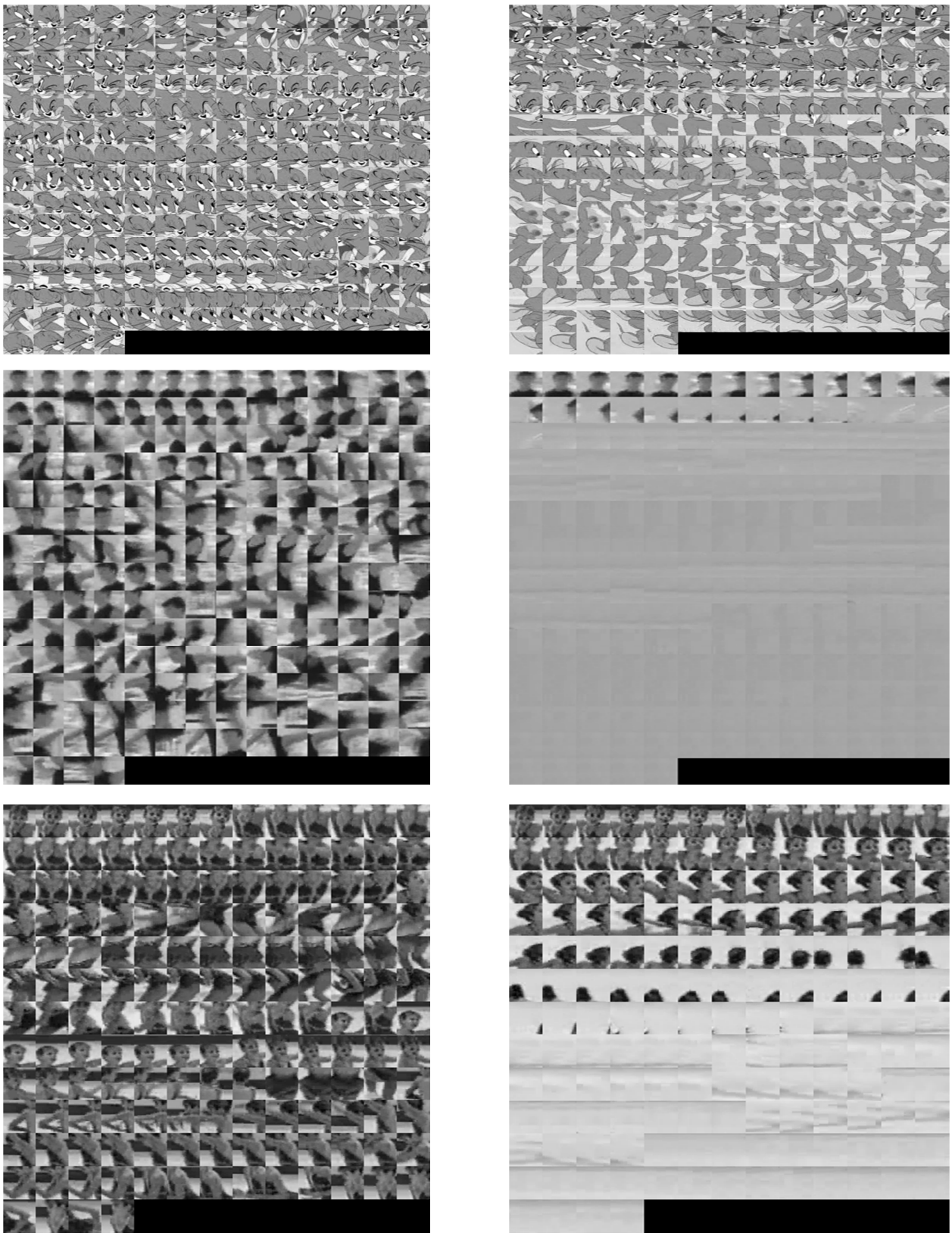


Figure 13: Performance of single block based matching using Bhattacharya distance, the left column indicates the use of initial template and the right column indicates use of a template update method.

4.2 SUB-BLOCK BASED MATCHING

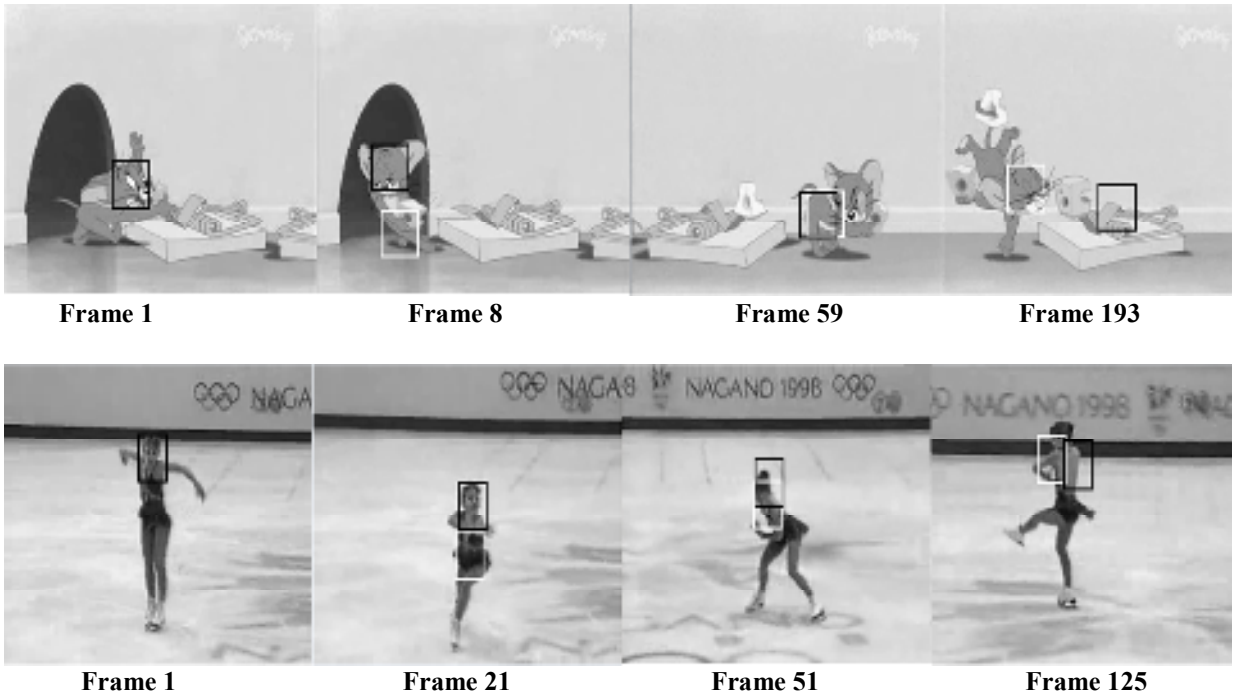


Figure 14: Results for sub-block based matching using Euclidean distance, the white block indicating the use of initial template and the black box indicating the use of a template update method.

It can be concluded that both the non-template update tracking and template update tracking are not consistent for both the cases. The template update tracking seems to be working up to a certain point, but when it loses the target due to some shape and appearance variations, the tracker does not recover. When the non-template update tracking is taken into consideration, the tracker is not that robust to pose changes and loses its target in some frames, but this tracker can recover as it uses a global template for matching in each and every frame.

Though this algorithm has shown satisfactory results, there are some deficiencies that it has not been able to overcome. Dividing the target into blocks arbitrarily may divide key features (hands, legs, chest etc.,) of the target into blocks. These features when considered as a whole may provide good information, rather than considering them in patches. This algorithm could not handle significant partial or full occlusions in the target very well.

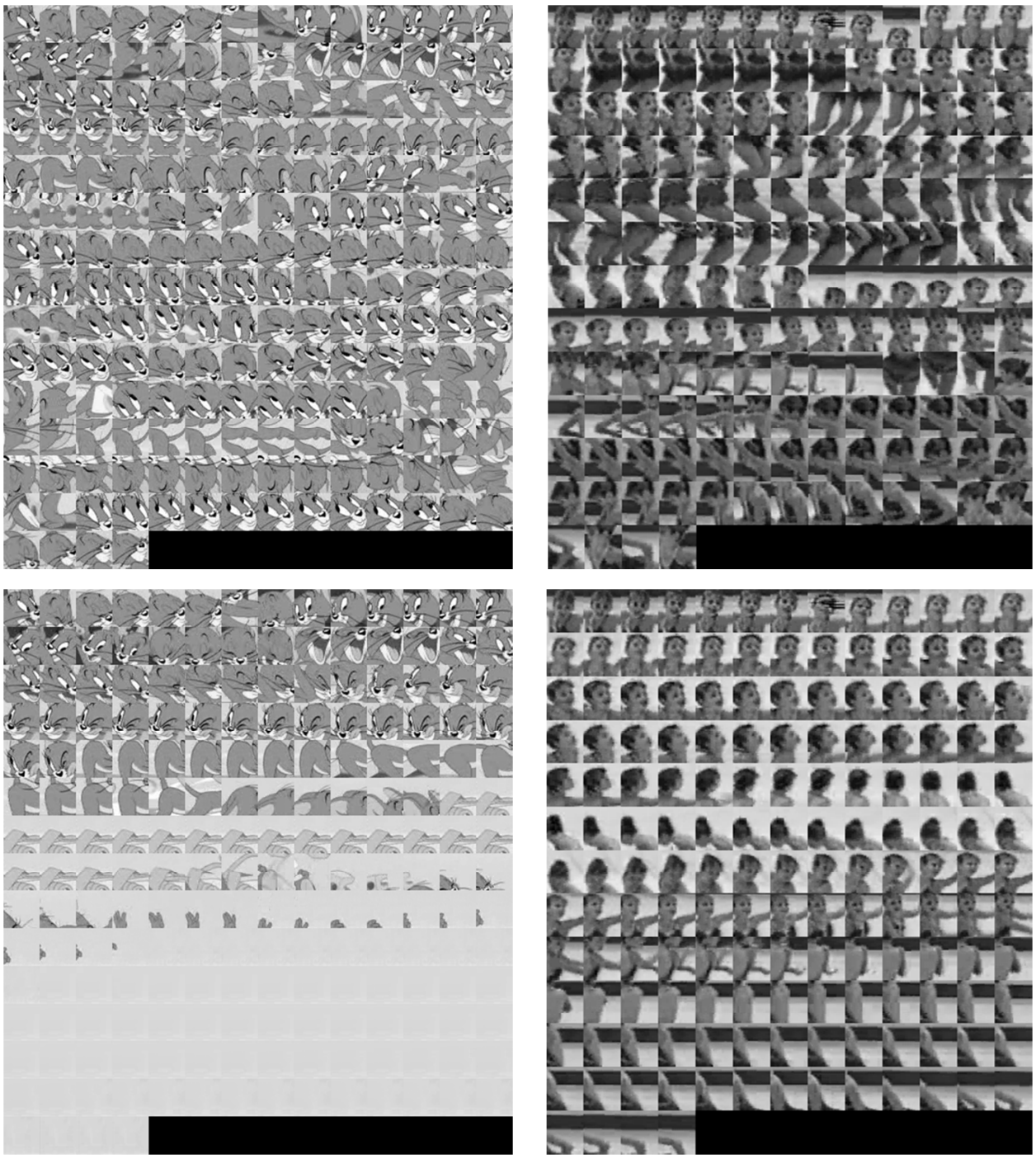
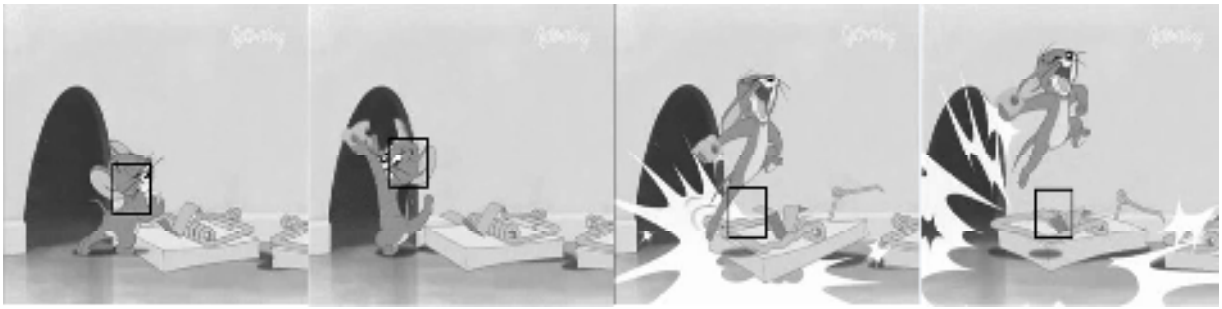


Figure 15: Performance of sub-block based matching using Euclidean distance, the top row indicates the use of initial template and the bottom row indicates the use of a template update method.

4.3 COVARIANCE TRACKING

4.3.1 LOCAL SEARCH

Because of the complexity of the algorithm, the time cost per frame increased by a considerable amount. To bring the time cost down, the algorithm is initially tested by limiting its search window to a 50 pixel radius.

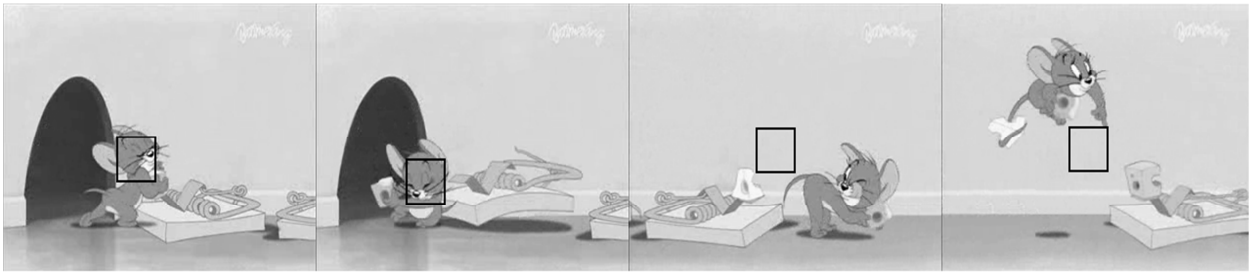


Frame 5 **Frame 16** **Frame 31** **Frame 33**

Figure 16: Results for covariance tracking with a 50 pixel search window.

From the above results, it can be observed that when there are sudden movements of the target, limiting the search window to a 50 pixel radius from the last known coordinates does not help. It tries to get the best match of the target in the search window and identifies it as the target. To overcome this low performance, an exhaustive search is employed on the entire window to observe the performance of the tracker, even though the time taken for each frame is a little high when compared to this method.

4.3.2 EXHAUSTIVE SEARCH



Frame 5 **Frame 20** **Frame 123** **Frame 167**

Figure 17: Results for covariance tracking using exhaustive search.

When an exhaustive search is employed to better observe the performance of covariance tracking algorithm, the above figure illustrates the result of the algorithm. Only in this case, it is observed and learnt that the covariance parameter does not work in frames that have smooth texture in them.

For better understanding, in the original paper the author used different properties to form a feature vectors which in turn are used for building the covariance matrix. As the calculation of different properties for each block is time consuming, these properties are replaced with histograms in this method. When a smooth region is taken into consideration and divided into blocks, all the feature vectors will be having the same histograms with negligible variations. When a covariance matrix is built from these feature vectors, it will be a zero matrix. Another aspect is that the Forstner distance only works for non-zero covariance matrices. Though the covariance matrix is independent of the size of the object, its computation is dependent on the number of feature vectors used. As the number of feature vectors used to describe an object increase, so does its computation time.

4.4 MULTIPLE BLOCK BASED APPROACH

4.4.1 EQUAL WEIGHTS

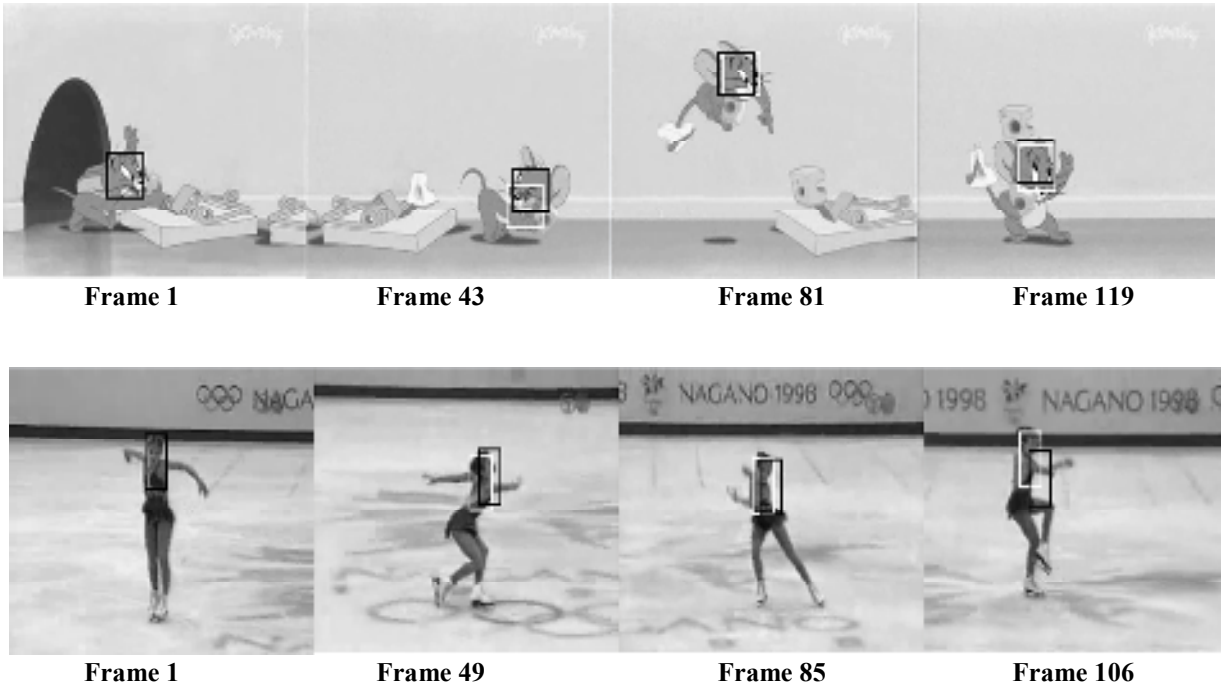
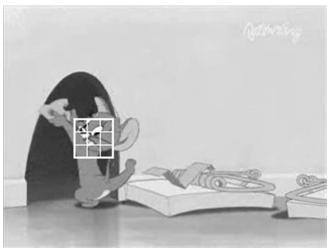


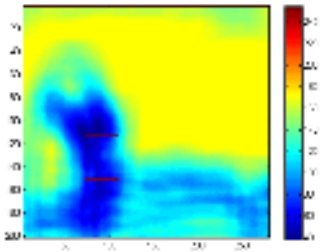
Figure 18: Results for multiple block based approach (equal weights), the white block indicating the use of initial template and the black box indicating the use of a template update method.

Above are the results of this algorithm, a slightly modified approach to the sub-block based approach. It can be observed that the updated template method is not giving good results when compared to the constant template method. To further experiment on this algorithm, the distances for each block are observed and to improve the performance of the tracking algorithm, some modifications are made.

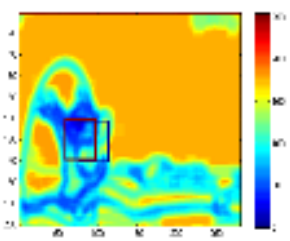
Frame 1



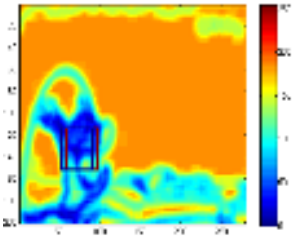
Frame 16



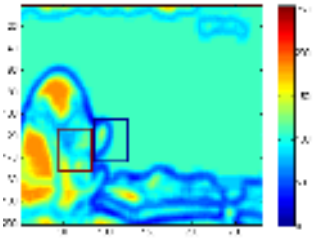
Overall distance map



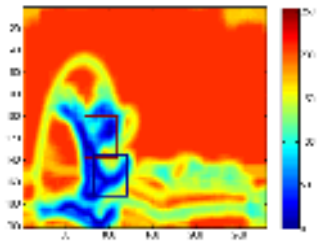
Block 1



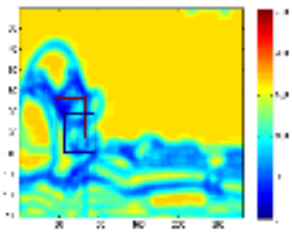
Block 2



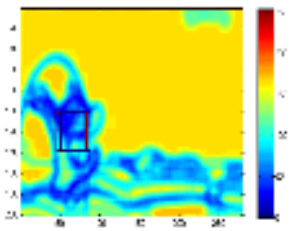
Block 3



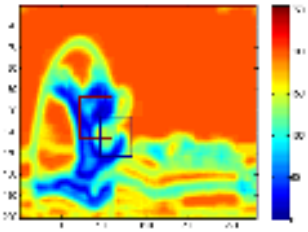
Block 4



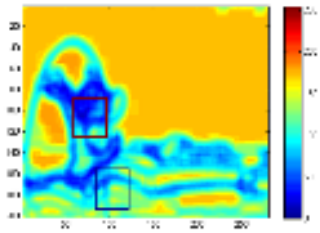
Block 5



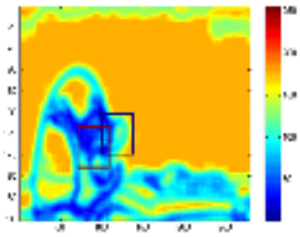
Block 6



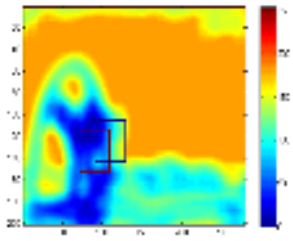
Block 7



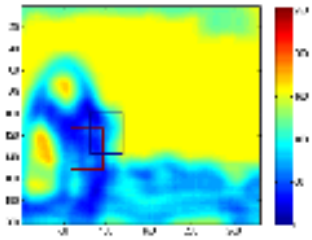
Block 8



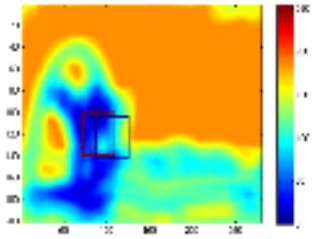
Block 9



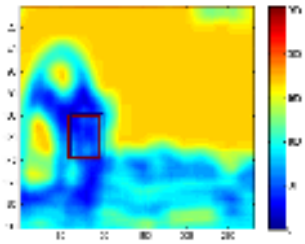
Block 10



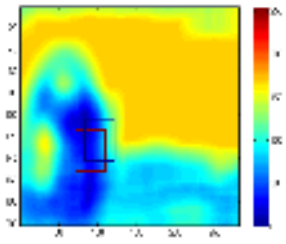
Block11



Block 12

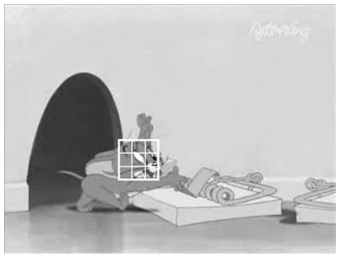


Block 13



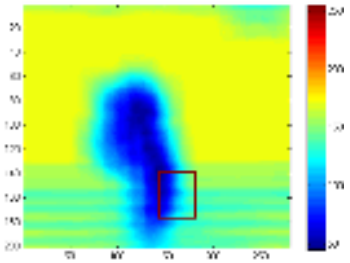
Block 14

Figure 19: Distance maps for each block for a good recognition using multiple block based approach, where the blue block indicates a local minima in each distance map and the red block indicates a global minima.

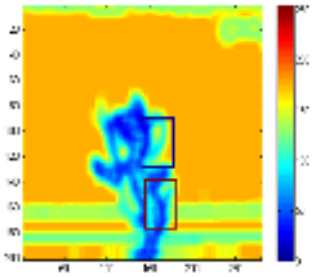


Frame 1

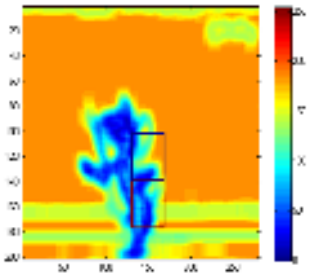
Frame 153



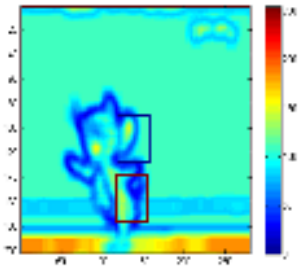
Overall distance map



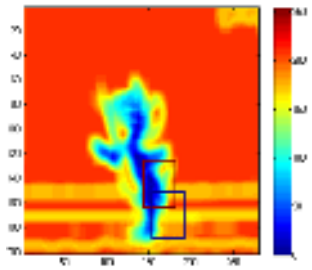
Block 1



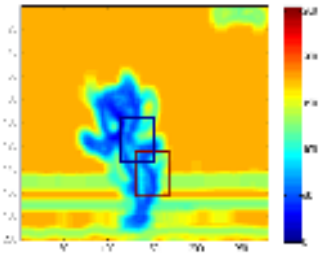
Block 2



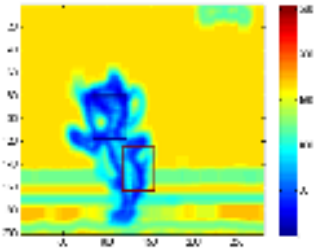
Block 3



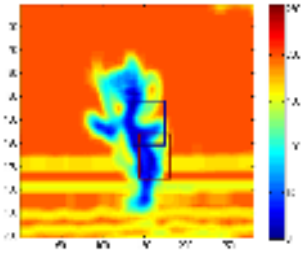
Block 4



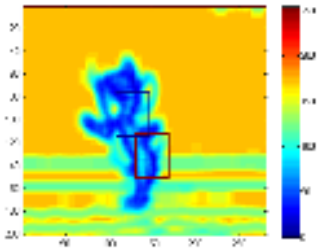
Block 5



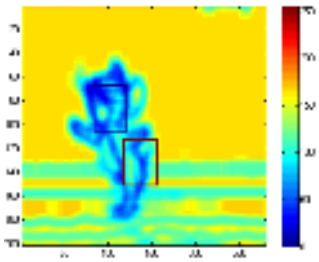
Block 6



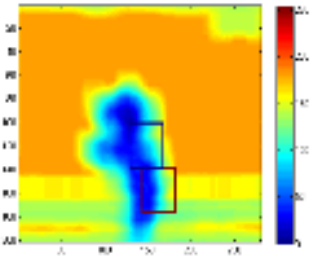
Block 7



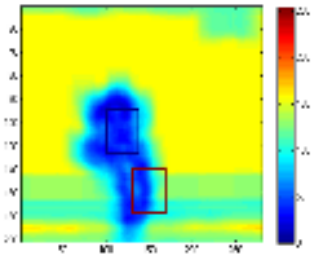
Block 8



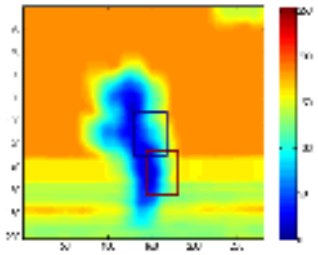
Block 9



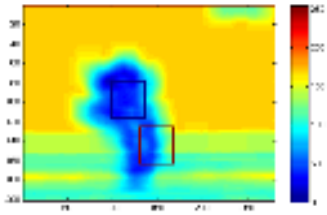
Block 10



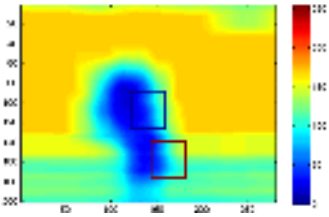
Block 11



Block 12



Block 13



Block 14

Figure 20: Distance maps for each block for a bad recognition using multiple block based approach, where the blue block indicates a local minima in each distance map and the red block indicates a global minima.

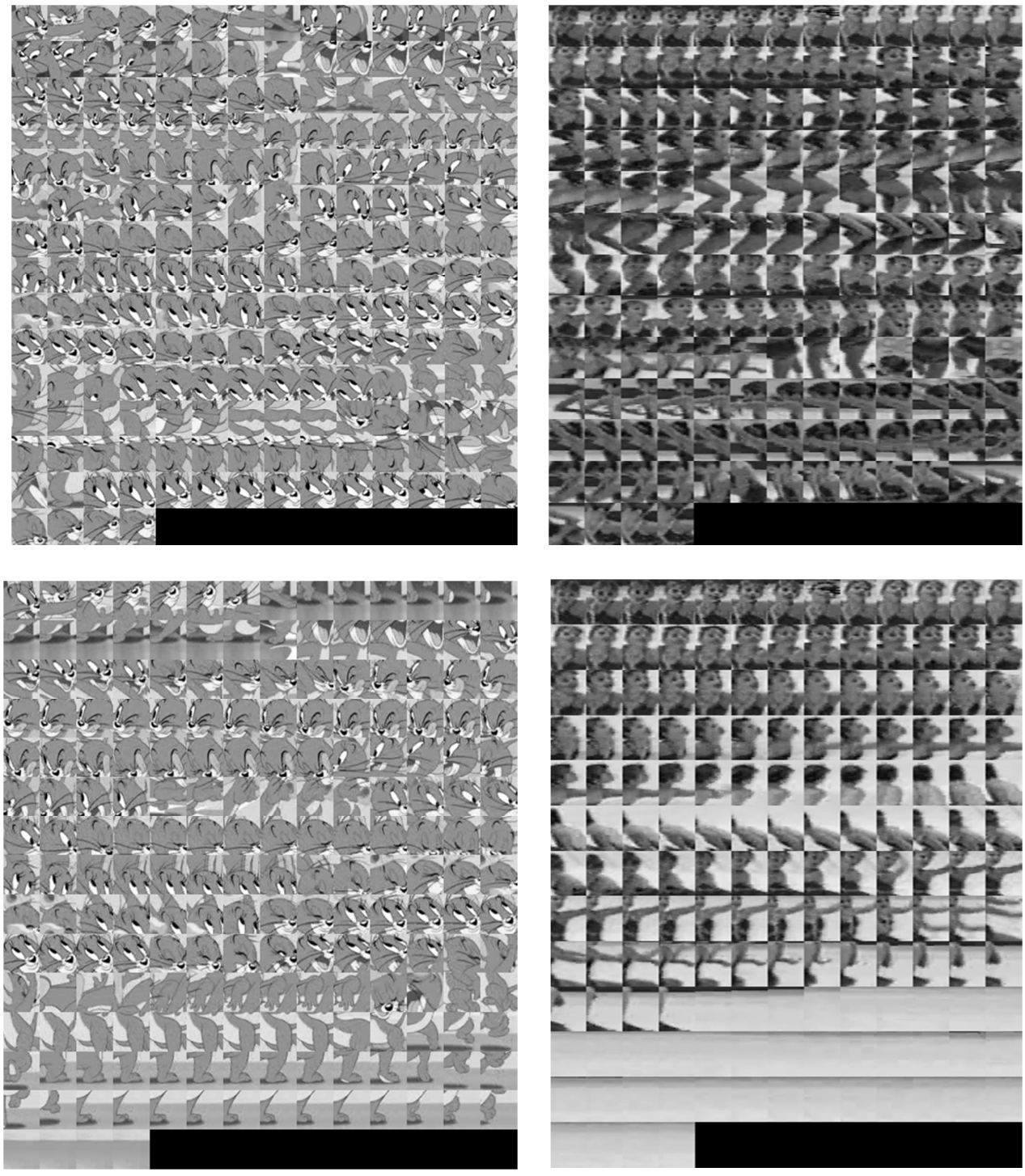


Figure 21: Performance of multiple block based approach (equal weights), the top row indicates the use of initial template and the bottom indicates the use of a template update method.

4.4.2 DIFFERENT WEIGHTS

One of the modifications made to this algorithm is giving weights to the blocks dependent on their sizes. In general, the original distance is an average of all the distances of the blocks. In this modification, weights are given to the blocks dependent on their size as in equation (9).

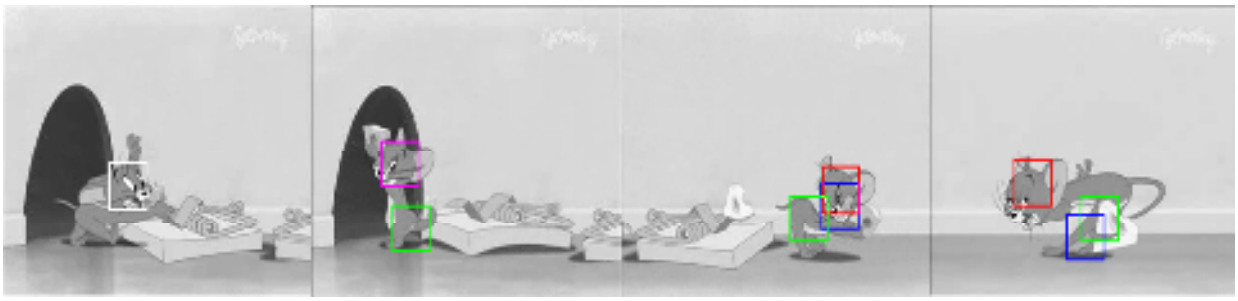
Here the weights to the blocks are taken depending on the size of the blocks. Three cases are considered, where weights are given proportional to the size of the blocks and also inversely proportional to the size:

Case 1: *Equal weights*

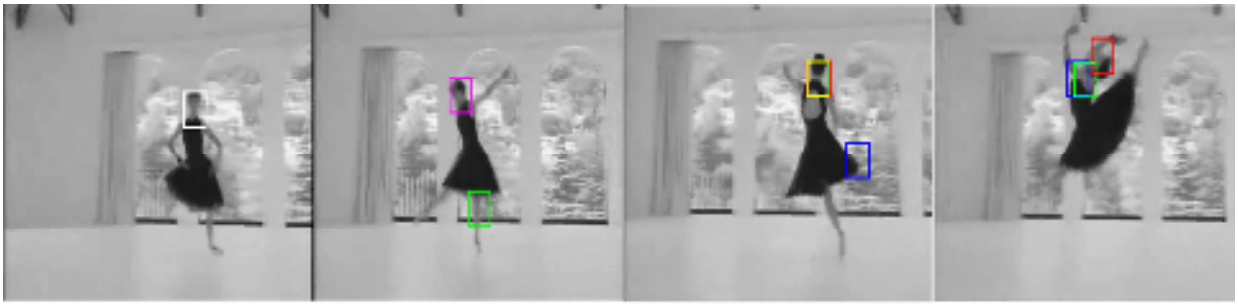
Case 2: w_1 *to* $w_9 = 1$, w_{10} *to* $w_{13} = 4$, $w_{14} = 9$

Case 3: w_1 *to* $w_9 = 1$, w_{10} *to* $w_{13} = 1/4$, $w_{14} = 1/9$

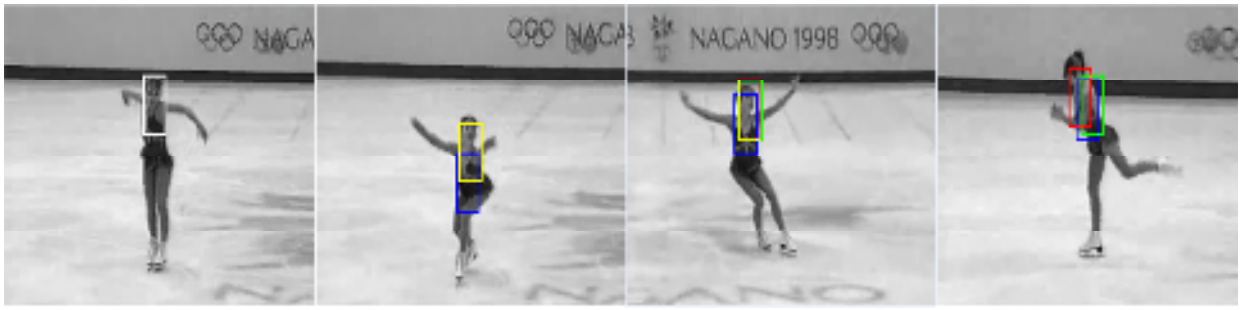
It can be observed from the figure that none of the cases performed consistently. Out of all the cases, the one that performed comparatively well is the one where equal weights are given to all the blocks. Different colors can be observed other than the red, green and blue colors, when there is an overlap of these colors.



Frame 1 Frame 18 Frame 58 Frame 143



Frame 1 Frame 17 Frame 88 Frame 128

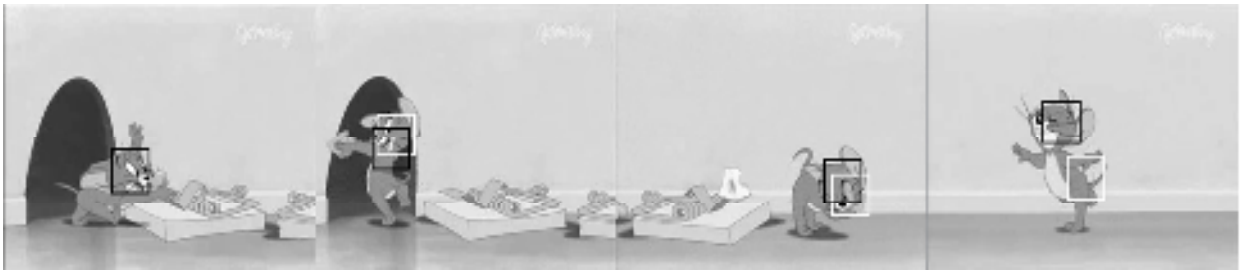


Frame 1 Frame 15 Frame 94 Frame 148

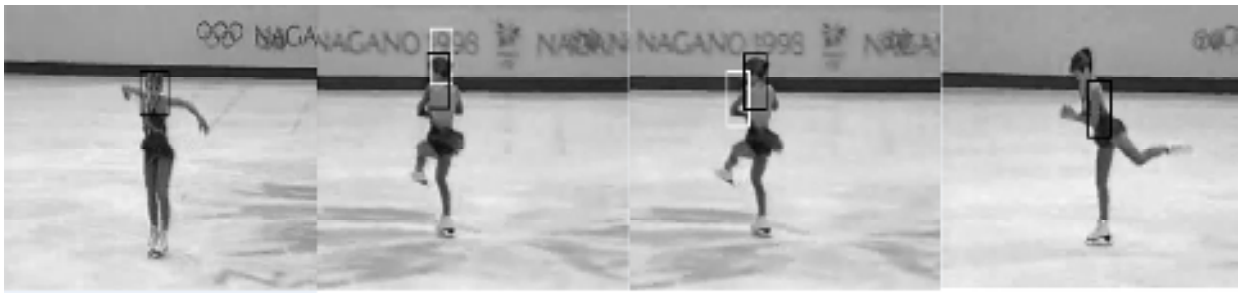
Figure 22: Results for multiple block based approach (different weights), Case 1: Red Block, Case 2: Blue Block, Case 3: Green Block.

4.4.3 THRESHOLDING THE DISTANCE

After observing the distance maps for all the blocks for each frame, there are some bad distance measures which had an effect on the entire distance measure. For limiting the effect of such bad distance measures, a threshold is imposed on very blocks similarity measure. Several thresholds are given to observe the reaction of the tracker.



Frame 1 **Frame 29** **Frame 73** **Frame 142**



Frame 1 **Frame 67** **Frame 73** **Frame 101**

Figure 23: Results for multiple block based approach (thresholding the distance), the black block indicating the results after using the threshold and the white block indicating the normal results.

After evaluating these results, there are some cases where this modification had given some results which are not better when compared to the normal results. When further evaluated, it was observed applying the threshold had some bad effects and that some good distance measure are being cut-off by using a threshold function. Finally, it was concluded that applying a threshold to the similarity measure had given some goods results when compared to the normal results but there are instances where the results are not better than the original results, which occurred due to thresholding some good similarity measures.

4.5 MOVING AVERAGE

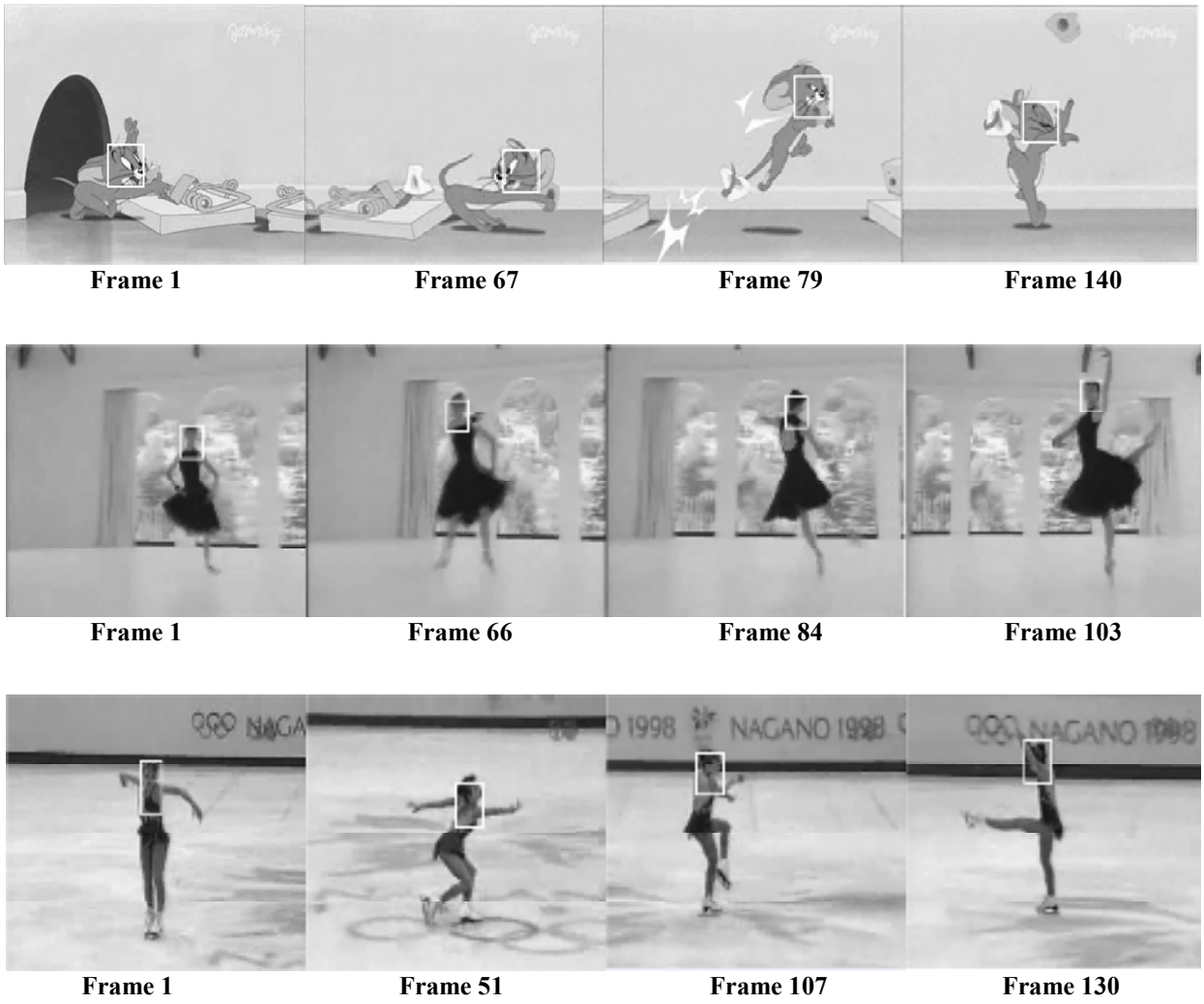
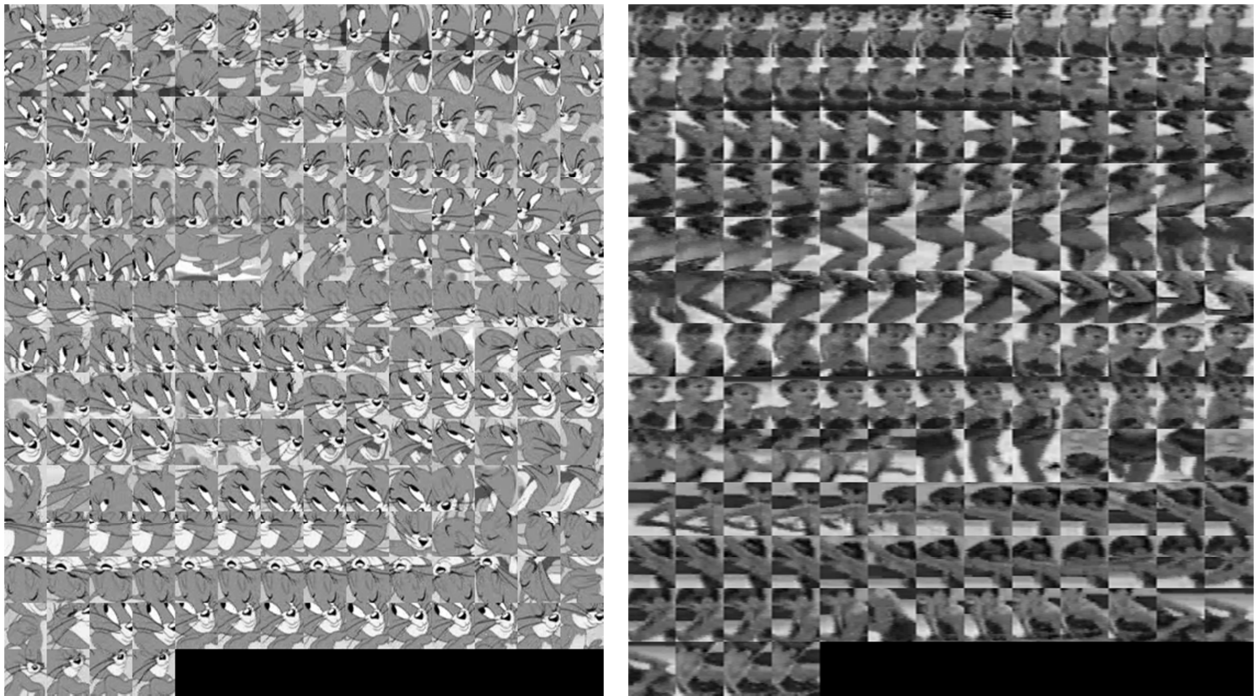


Figure 24: Results for moving average algorithm.

For the moving average algorithm in all the cases, the results are good when compared to the previous methods. Using an average of the last *M* recognized targets as a key feature for the matching function, did not allow any sudden changes in the tracking function. Although, this method comes under template update algorithm, the results are as consistent as that of constant template method, other than few minor defects. These results are better than all the template update models of all the previous methods.



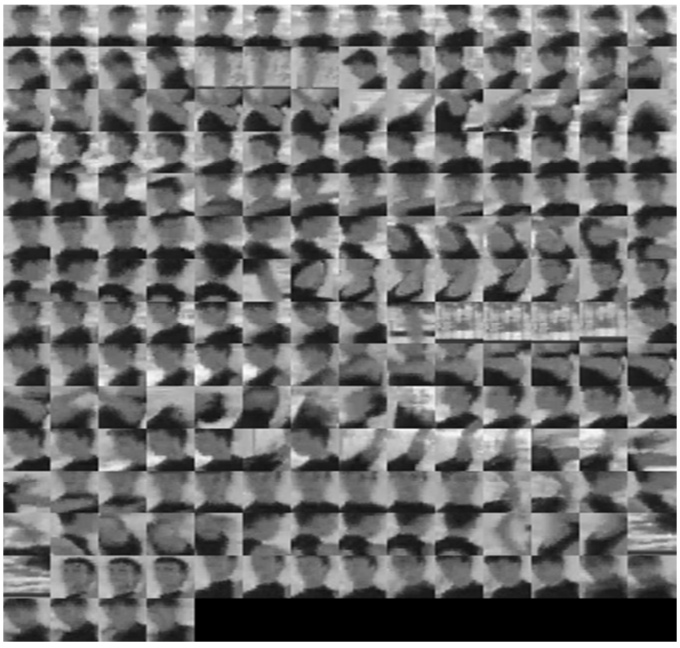


Figure 25: Performance of moving average algorithm.

CHAPTER 5: PERFORMANCE AND TIME COMPLEXITY

Dataset 1 – Cartoon (200 frames), Dataset 2 – Dancer (200 frames), Dataset 3 – Female Skater (160 frames).

Table 1: Tracking performance of the algorithms

Algorithm	Dataset 1	Dataset 2	Dataset 3
Single block based matching algorithm			
a) Euclidean distance			
• Initial Template	85%	80%	81.3%
• Template Update	87.5%	65%	84.4%
b) Bhattacharya distance			
• Initial Template	85%	75%	75%
• Template Update	40%	15%	43.8%
Sub-block based algorithm			
• Initial Template	90%	87.5%	81.3%
• Template Update	35%	40%	62.5%
Multiple block based algorithm			
• Initial Template	92.5%	87.5%	84.4%
• Template Update	80%	85%	62.5%
Moving average algorithm	95%	87.5%	81.3%

All the datasets, the algorithms are experimented on, are gray scale images of size 320x240. The percentages in the Table 1 indicate the percentage of good recognitions in each approach, and they indicate the tracking performance of each algorithm and not accuracy. A number of datasets were considered for evaluating the performance of these algorithms, but these three datasets were given importance as they had significant shape and pose variations when compared to the others, as they could evaluate the algorithms performance better.

Initially, all the algorithms are implemented in MATLAB to observe the performance of the algorithms and the effect of the improvements. After observing the results, based on their performance the Matlab code is replaced with C files for computational time analysis. For every algorithm, most of the code is implemented using C, other than Matlab commands like imread, double, imshow, imwrite. These C files are compiled and used in Matlab using a Mex function.

All the 'integral histogram' based algorithms were run on a system with specification: Intel Xenon 64 bit Processor @ 2.4 GHz with RAM 4.00 GB. Initially, the integral histogram function is run on a standard 512x512 gray scale image which took 60 msec. The experimental datasets that are used for the analysis of the algorithms performance are all of 320x240 gray scale images. For all the approaches, the number of bins used is constant i.e., $N = 16$. The computation time for a single frame in case of each algorithm is as follows:

Table 2: Computation time for each algorithm

Algorithm	**Time taken per frame**
Single block based approach a) Euclidean distance b) Bhattacharya distance	 33 msec 35 msec
Sub-block based approach	75 msec
Multiple block based approach	90 msec
Moving average approach	95 msec

CHAPTER 6: CONCLUSION AND FUTURE WORK

The integral histogram algorithm certainly has taken the histogram-based tracking algorithms to a new level. In the traditional histogram based tracking algorithms, tracking by using an exhaustive search on the entire frame takes ample amount of time, since histogram retrievals of multiple blocks takes considerable time. By using integral histogram, these histogram retrievals of blocks can be computed by some simple arithmetic operations, independent of the size of the block. This algorithm has increased the number of features that can be incorporated in a feature vector of the target that is used in the matching function and has given some amount of freedom for the user to develop a good feature vector that represents the target, which can increase the performance of the visual tracker.

As stated earlier, a number of algorithms were proposed, which have used the integral histogram function in their approaches, made an attempt to use the integral histogram in a more efficient manner. A number of these algorithms are implemented in this paper, to further study the effect of the approaches used in them. A comparison was also made among these algorithms to observe which algorithm had the best utilization for the integral histogram function and better serve the purpose of the real time applications.

From the tracking results that are obtained from these algorithms after being implemented, the algorithms that had a better performance when comparatively are the multiple block based approach and the moving average algorithm. Although there are some defects that are to be taken care of, these two algorithms gave some promising results and could track at approximately 10 frames per second.

A number of possible directions for future research can be stated after studying the performance of these algorithms. Foremost among them is search for an efficient algorithm, which could give weights to the blocks depending upon the foreground and background pixels present in the block and updating these weights at every frame, after the possible target location is recognized. Use of both spatial and statistical properties of the target to provide an elegant feature vector that can represent the target much better, could be useful to improve the performance of the tracker.

Another area of possible research is the use of template update versus initial template, which can be a more efficient method to deal with problems like target occlusions, noise, rapid shape and pose variations, illumination conditions. Another possible direction is developing a tracking algorithm that makes a robust use of both the initial template and the last recognized template in the construction of feature vector of the target. This approach could take care of several problems the present object tracking algorithms are facing. An efficient use of the integral histogram can also be made by fusing multiple features into the feature vector, which could give a better description of the region of interest. These features can include spatial properties, intensity or color histograms, local color distributions, prominent features of the object with respect to the tracking task.

REFERENCES

[1] M. Krinidis, N. Nikolaidis, I. Pitas, "2D Feature Point Selection and Tracking Using 3D Physics-Based Deformable Surface," *IEEE Trans. on Circuits and Systems for Video Technology (CSVT),* 2007.

[2] J. Shao, S. K. Zhou, R. Chellappa, "Appearance-Based Tracking and Recognition Using the 3D Trilinear Tensor", 2004.

[3] B. Han, W. Roberts, D. Wu, J. Li, "Robust Feature-based Object Tracking," *SPIE Defense & Security Symposium,* 2007.

[4] Y. Rathi, N. Vaswani, A. Tannenbaum, A. Yezzi, "Particle Filtering for Geometric Active Contours with Application to Tracking Moving and Deforming Objects," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI),* 2007.

[5] A. Yilmaz, X. Li, M. Shah, "Contour-Based Object Tracking with Occlusion Handling in Video Acquired Using Mobile Cameras," *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI),* 2004.

[6] M. Yazdi, M. Seyfi, A. Rafati, M. Asadi, "Real-time Lip Contour Tracking For Audio-Visual Speech Recognition Applications," *International Journal of Biological and Life Sciences,* 2008.

[7] S. Fazli, H. M. Pour, H. Bouzari, "Multiple Object Tracking Using Improved GMM Based Motion Segmentation," *IEEE Conf. on Electrical / Electronics Engineering, Computer and Information Technology (ECTI),* 2009.

[8] C. Yang, R. Duraiswami, L. Davis, "Efficient Mean-Shift Tracking via a New Similarity Measure," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2005.

[9] Z. Li, Q. Tang and N. Sang, "Improved Mean Shift Algorithm for Multiple Occlusion Target Tracking", *IEEE Trans. on Pattern Recognition & Artificial Intelligence,* 2008.

[10] G. D. Hager, M. Dewan, C. V. Stewart, "Multiple Kernel Tracking with SSD," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2004.

[11] F. Porikli, O. Tuzel, "Multi-Kernel Object Tracking", *IEEE International Conf. on Multimedia and Expo (ICME)*, 2005.

[12] Y. Boykov, D. P. Huttenlocher, “Adaptive Bayesian Recognition in Tracking Rigid Objects,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2000.

[13] G. Welch, G. Bishop, “An Introduction to Kalman Filter,” 2006.

[14] G. Klein and D. Murray, “Full-3D Edge Tracking with a Particle Filter,” 2006.

[15] F. Porikli, “Integral Histogram: A Fast Way to Extract Histograms in Cartesian Spaces,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2005.

[16] A. Adam, E. Rivlin and I. Shimshoni, “Robust Fragments-based Tracking using the Integral Histogram,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2006.

[17] S. M. Nejhum, J. Ho, M. H. Yang, “Online Visual Tracking with Histograms and Articulating Blocks,” *IEEE Conf. on Computer Vision and Image Understanding*, 2010.

[18] F. Porikli, O. Tuzel, P. Meer, “Covariance Tracking using Model Update Based on Lie Algebra,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2006.

[19] W. Forstner and B. Moonen, “A Metric for Covariance Matrices,” *Stuttgart University,* 1999.

[20] G. Hager, P. Belhumeur, “Efficient region tracking with parametric models of geometry and illumination,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1998.

[21] T. Müller, C. Lenz, S. Barner, A. Knoll, “Accelerating Integral Histogram using an Adaptive Approach,” *IEEE International Conf. on Image and Signal Processing*, 2008.

[22] S. Avidan, “Ensemble Tracking,” *IEEE International Conf. on Computer Vision and Pattern Recognition (CVPR),* 2005.

[23] O. Tuzel; F. Porikli; P. Meer, "Region Covariance: A Fast Descriptor for Detection and Classification," *European Conf. on Computer Vision (ECCV),* 2006.

[24] D. Comaniciu, V. Ramesh, P. Meer, “Real Time tracking of Non-rigid Objects Using Mean shift,” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 2002.

[25] P. Viola, M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* 2004.

VITA

Sriharsha Atluri was born in Guntur, Andhra Pradesh (India), in 1987. He received a bachelor's degree in electronics and communication engineering at the Jawaharlal Nehru Technological University, Hyderabad, in May 2008. He started his work towards the degree of master's in engineering science at Louisiana State University in fall 2008. He worked as a research assistant during his time as a master's student under Dr. Bahadir K. Gunturk in the Department of Electrical Engineering.