Visual Tracking using Analysis Dictionary Learning

Submitted by

Divya Sitani

under the guidance of

Dr. A. V. Subramanyam

Asst. Professor, IIIT-Delhi

and

Dr. Angshul Majumdar

Asst. Professor, IIIT-Delhi

Indraprastha Institute of Information Technology
New Delhi
August 2017

Visual Tracking using Analysis Dictionary Learning

By
Divya Sitani

Submitted
in partial fulfilment of the requirements for the degree of Master of Technology

to

Indraprastha Institute of Information Technology Delhi
August, 2017

Certificate

I hereby certify that the thesis titled, **Visual Tracking using Analysis Dictionary Learning** being submitted by **Divya Sitani** to Indraprastha Institute of Information Technology Delhi, for the award of Master of Technology, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree.

August 2017

Dr. A. V. Subramanyam and Dr. Angshul Majumdar
Department of Electronics and Communication
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

Acknowledgement

First of all, I would like to thank my guide Dr. A. V. Subramanyam for his constant support and encouragement. He has been extremely patient the entire time. He had complete faith in my ideas and helped me implement my ideas with all his guidance and support.

I would like to express my sincere gratitude to Dr. Angshul Majumdar. His help at various stages during this work right from the first day has been extremely valuable. He patiently cleared all my doubts and confusions and eventually helped me get a better understanding of my work.

I would like to thank Mr. Shishir Sharma for all his help and support throughout this work.

I am grateful to the institute for all its infrastructure support and for providing such a conducive environment to young people to be able to think differently. I am also thankful to all my seniors and colleagues who helped me in various ways throughout the thesis.

I dedicate this thesis to my mother who has always been an unending source of inspiration and love for me. She has always trusted my abilities and given me the strength to believe in myself.

Finally, I wish to thank my sister and my father for always having faith in me and standing strong with me in all the ups and downs of life.

Divya Sitani

Contents

List of Figures

List of Tables

Abstract

Visual tracking or object tracking is the process of estimating the state of the target in successive frames of a video sequence. It is an integral part of a plethora of applications like security, surveillance, navigation systems, traffic monitoring systems, human computer interaction systems, and robotics where the target is tracked in both stationary as well as dynamic environments. Visual tracking has remained a challenging problem in computer vision because of numerous factors like occlusion, illumination variation, background clutter, pose change, scale variation, deformation, etc. To overcome these challenges we propose an online analysis dictionary learning framework for visual tracking.

Dictionary learning is a popular representation learning tool today. It has been successfully applied to a wide range of computer vision tasks like image denoising, image super resolution, face recognition, human action recognition, classification, etc. in the recent years. Synthesis dictionary based learning approaches have been applied to visual tracking as well. However, to the best of our knowledge, the use of analysis dictionary based algorithms for visual tracking has not been done yet. The main advantage of an analysis dictionary over a synthesis dictionary is, for an analysis and a synthesis dictionary of same dimensions, an analysis dictionary is able to capture significantly more variability in the data compared to a synthesis dictionary.

We have developed our algorithm in two stages. In this first stage we track the targets in video sequences using a single analysis dictionary. In the next stage we develop a multiple analysis dictionaries model to track the object of interest. After extensive experimentation on video sequences from OTB-50 dataset, we have demonstrated that our algorithm works better than the synthesis dictionary learning based trackers and also some of the other state of the art trackers that do not incorporate dictionary learning in their tracking approach.

Chapter 1

Introduction

Visual object tracking is the process of predicting the future state of the target in an image sequence given the state of the target in the first frame. The main challenges to overcome while tracking can be divided into two categories, intrinsic variations and extrinsic variations. The intrinsic variations are the appearance changes that the target goes through in the subsequent frames like scale variation, deformation, blurring in the target region due to fast motion of the camera or target, in and out of plane rotations of the target in the image plane, fast motion of the target, low resolution in the target region and a part of the target goes out of view from the frame. Extrinsic variations are the transitions that occur in the background of the target like illumination variation, occlusion and background clutters.

A reliable tracking algorithm should be able to fulfil two requirements: it should be able to adapt to the intrinsic variations that occur in the target and it should be invariant to the extrinsic variations that occur in the background of the target, as the sequence progresses. In our work we propose two discriminative trackers to deal with these variations that affect the tracking process. Discriminative tracking algorithms solve a binary classification problem of discriminating the foreground or the target from its background. Tracking algorithms have an appearance model which represents the visual characteristics of the target by extracting feature information from the target region.

In recent decades, implementations in which the appearance model is based on dictionary learning have become quite popular [1–9]. These tracking algorithms that incorporate dictionary learning in their framework use synthesis dictionary learning based approaches. In the synthesis model, a signal $x \in \mathbb{R}^n$ can be sparsely represented in a dictionary $D \in \mathbb{R}^{n\times k}$, as $x = Dz$, where $z \in \mathbb{R}^k$ is sparse, i.e. $\|z\|_0 \ll k$. In our work we have used analysis dictionary learning to learn our dictionary. In the analysis model, a signal x is sparsifiable using a dictionary $D \in \mathbb{R}^{m\times n}$, that is $Dx = z$, where $z \in \mathbb{R}^m$ is sparse.

A little analysis shows that for a synthesis dictionary of size $m \times n$, with sparsity(number of non-zero elements in the sparse code z) being equal to k, the number of sub-spaces is $\binom{n}{k}$ and each subspace is $k-$ dimensional. For analysis dictionary learning of size $p \times d$, with co-sparsity l, the number of sub-spaces is $\binom{p}{l}$ for sub-spaces of dimension $d - l$. In order to compare the synthesis and analysis model, let us consider the dimensions of the subspaces are same, i.e., $k = d - l$. Also let us assume equal redundancy, i.e. $p = n = 2d$. Then the number of synthesis subspaces available are $k.log_2\left(\frac{n}{k}\right)$ (by Stirling's Approximation) and the number of available analysis subspace are n.

For example with $n = 700$, $l = 300$ and $k = 50$, the number of analysis sub-spaces are 700 whereas the number of synthesis sub-spaces are only 191. This means that for an analysis and a synthesis dictionary of same dimensions, an analysis dictionary is able to capture significantly more variability in the data compared to a synthesis dictionary. A more detailed analysis can be found in [10].

Further, in the second stage of our algorithm we use multiple analysis dictionaries for tracking. Inspired by [9] we use a feature weighting scheme, where each feature gets a weight in the appearance model based on its capability to describe the data, the feature which has higher capability to describe the data gets a higher weight as compared to other features. Some approaches have earlier used multiple dictionaries [11], but they do not consider the fact that different features should have different importance.

1.1 Visual Tracking Background

In this section we discuss several existing popular tracking algorithms. Our work using single dictionary is inspired by [1]. Thus, a description of these popular trackers and [1] is as follows.

The tracking algorithm in [12] MDNET, is based on the representations from a discriminatively trained Convolutional Neural Network (CNN). The algorithm pretrained a CNN using a large set of videos with tracking ground-truths to obtain a generic target representation. The network is composed of shared layers and multiple branches of domain-specific layers, where domains correspond to individual training sequences and each branch is responsible for binary classification to identify the target in each domain. The authors train the network with respect to each domain iteratively to obtain generic target representations in the shared layers. When tracking a target in a new sequence, they construct a new network by combining the shared layers in the pretrained CNN with a new binary classification layer, which is updated online. Online tracking is performed by evaluating the candidate windows randomly sampled around the previous target state.

In [13], tracking algorithm MEEM (Robust Tracking via Multiple Experts using Entropy Minimization) is proposed. It is a multi-expert restoration scheme to address the model drift problem in online tracking. In the scheme, a tracker and its historical snapshots constitute an expert ensemble, where the best expert is selected to restore the current tracker when needed based on a minimum entropy criterion, so as to correct undesirable model updates. The base tracker in the formulation exploits an online SVM on a budget algorithm and an explicit feature mapping method for efficient model update and inference.

In [14], tracking algorithm DSST (Discriminative Scale Space Tracker) is proposed. It is an approach which works by learning discriminative correlation filters which is based on a scale pyramid representation. The authors learn separate filters for translation and scale estimation. They show that this improves the performance compared to an exhaustive scale search. They propose that their scale estimation approach is generic as it can be incorporated into any tracking method with no inherent scale estimation.

In [15], tracking algorithm TGPR analyses the probability of target appearance using Gaussian Processes Regression (GPR), and introduce a latent variable to assist the tracking decision. They observation model for regression is learnt in a semi-supervised fashion by using both labelled samples from previous frames and the unlabelled samples that are tracking candidates extracted from the current frame.

In [16], the tracking algorithm KCF (Kernelized Correlation Filters) proposes an analytic model based on discriminative tracking for datasets of thousands of translated patches. By showing that the resulting data matrix is circulant, the authors diagonalize it with the Discrete Fourier Transform, reducing both storage and computation by several orders of magnitude. For linear regression their formulation is equivalent to a correlation filter. For kernel regression, they derive a new Kernelized Correlation Filter (KCF), that has the exact same complexity as its linear counterpart.

In [17], the tracking algorithm STRUCK (Structured Output Tracking with Kernels) proposes a framework for adaptive visual object tracking based on structured output prediction. The method uses a kernelized structured output support vector machine (SVM), which is learned online to provide adaptive tracking. To allow for real-time application, the authors introduce a budgeting mechanism which prevents the unbounded growth in the number of support vectors which would otherwise occur during tracking.

In [8], the authors have proposed a robust object tracking algorithm SCM (Sparsity based collaborative model) using a collaborative model. Their appearance model exploits both holistic templates and local representations. They develop a sparsity-based discriminative classifier (SDC) and a sparsity-based generative model (SGM). In the SDC module, they introduce a method to compute the confidence value that assigns more weights to the foreground than the background. In the SGM module, they proposed a histogram-based method that takes the spatial information of each patch into consideration.

In [18], a tracking framework TLD (Tracking-Learning-Detection) is proposed that explicitly decomposes the long-term tracking task into tracking, learning and detection. The detector localizes all appearances that have been observed so far and corrects the tracker if necessary. The learning estimates detector's errors and updates it to avoid these errors in the future. The authors identify detector's errors and learn from them.

In [1], authors have proposed a tracking framework based on sparse representation and online discriminative dictionary learning (ODDL). They associate dictionary items with label information, such that the learned dictionary is both reconstructive and discriminative, which distinguishes target objects from the background. During tracking, the best target candidate is selected by a joint decision measure. Reliable tracking results and augmented training samples are accumulated into two sets to update the dictionary. Both online dictionary learning and the proposed joint decision measure are important for the final tracking performance.

In the next chapter, we explain the problem formulation, initialization, classification and the tracking procedure of both our proposed algorithms.

Chapter 2

Proposed Algorithms

We propose our algorithm in two stages. In the first stage we have used single analysis dictionary learnt using HoG (Histogram of Oriented Gradients). In the second stage we perform feature fusion by using a weighting scheme learned on multiple dictionaries learnt using HoG (Histogram of Oriented Gradients) and LDP (Local Derivative Patterns). This multiple dictionary model is used as the appearance model in the second stage of the algorithm.

2.1 Tracking using single Analysis Dictionary

In the first stage we use a single analysis dictionary and hence, the name of our algorithm is **OADL**, Online analysis Dictionary Learning. The dictionary is learnt using HoG (Histogram of Oriented Gradients) features [19], as the appearance model. The problem formulation is explained below:

2.1.1 Problem Formulation

We have a set of training samples $X = \{x_1, x_2, ..., x_n\}$ as input and $X \in \mathbb{R}^{d \times n}$. Here, d is the dimension of HoG vector. The value of d is 496. Now our aim is to learn a dictionary on input training samples, that helps to solve a binary classification problem of discriminating foreground from the background. Therefore, the class labels are $Y = \{1, -1\}$. Here, $label = 1$ corresponds to the foreground or the target and $label = -1$ indicates background or the surrounding environment of the target. Each x_i is a feature vector extracted from a 32x32 image patch using histogram of gradients for feature representation. This image patch corresponds to either the positive sample which is our tracking target or a negative sample which is the background in a frame of a video sequence.

Let the given dictionary be $D = \{d_1, d_2, ..., d_k\} \in \mathbb{R}^{k\times d}$ and k is the number of atoms. The sparse code corresponding to each x_i is z_i. In the analysis formulation each $z_i \approx Dx_i$ and $z_i \in \mathbb{R}^{k\times 1}$. The set of all sparse codes combined together is $Z = \{z_1, z_2, ..., z_n\}$. Thus, $Z \approx DX$ and $Z \in \mathbb{R}^{k\times n}$. The loss function over D and Z is:

$$min_{D,Z} \|DX - Z\|_F^2 + \lambda_2 \|D\|_F^2 + \lambda_1 \|Z\|_1 \tag{2.1.1}$$

The term $\|DX - Z\|_F^2$ in Equation (2.1.1) is called the sparsification error for the data X in the dictionary domain. It is the modelling error in the dictionary model. Our aim is to minimize this error in order to learn the best analysis dictionary model. Here λ_1 and λ_2 are regularization parameters.

After computing D and Z, we find classifier W on the value of Z that has been calculated using Equation (2.1.1). Thus, we are solving two independent loss functions. For formulating the loss function for calculating classifier W, we proceed as follows.

$$min_W \|F - WZ\|_F^2 + \lambda_3 \|W\|_F^2 \tag{2.1.2}$$

The term $\|F - WZ\|_F^2$ in the above equation is a linear regression loss or the classification error. Each column of F, which is f_i, is a label vector for x_i and $f_i \in \mathbb{R}^m$. So, $f_i = [1, 0]^T$ indicates a positive sample or class label$= 1$ and $f_i = [0, 1]^T$ indicates a negative sample or class label $= -1$. The classifier $W \in \mathbb{R}^{m\times k}$. Number of rows in the classifier, $m = 2$, because there are two classes, one positive and the other negative

2.1.2 Initialization

KSVD [20] is run on positive and negative samples separately to form two dictionaries of the same size. These two are combined together to form a dictionary $D_{init} \in \mathbb{R}^{d \times k}$. Now we assume our initial analysis dictionary D_0 is the transpose of D_{init}. Thus, $D_0 = (D_{init})^T$ and $D_0 \in \mathbb{R}^{k \times d}$. The label f_i for each dictionary atom remains same in subsequent learning, only the value of each atom d_i is updated. Once we have the initial D_0, the sparse code is calculated over the initial dictionary:

$$Z_{calc} = argmin_Z \|DX - Z\|_F^2 + \lambda_1 \|Z\|_1 \tag{2.1.3}$$

Z is calculated using soft thresholding with λ_1 as the threshold. Here, $D = D_0$. Now, once we have the initial dictionary D_0 and the sparse code Z_{calc}, we have to compute the initial classifier W_0. W_0 is computed using ridge regression,

$$W_0 = argmin_W \|F - WZ_{calc}\|_F^2 + \lambda_3 \|W\|_F^2 \tag{2.1.4}$$

Here, F is the label matrix as mentioned in Section (2.1.1). The solution to Equation (2.1.4) is:

$$W_0 = FZ_{calc}^T (Z_{calc} Z_{calc}^T + \lambda_3 I)^{-1} \tag{2.1.5}$$

Now we have both the initial classifier and the initial dictionary.

2.1.3 Classification

For every new frame, a total of P particles are sampled around the centroid of the previous frame. Now we extract the feature vector x_i from a 32x32 image patch around the i^{th} particle, for each of the P particles. The feature vector x_i is extracted from the image patch by finding a 496 dimensional histogram of gradients feature using the toolbox provided by [21]. This is how we obtain a new data matrix $X = \{x_1, x_2, ...x_P\}$. We find the sparse code Z of X using Equation (2.1.3). Then a joint decision metric is calculated for each particle during testing. For calculating this joint decision metric, we first find out reconstructed x_i ($x_{i(rec)}$) using the sparse code z_i and dictionary D. $x_{i(rec)}$ and joint decision error metric over each particle are:

$$x_{i(rec)} = (D^T D + \lambda_1 I)^{-1} D^T z_i \tag{2.1.6}$$

$$\varepsilon(x_i) = \alpha \|X_{(tr)} - x_{i(rec)}\|^2 + (1-\alpha)\|f_i - W z_i\|^2 \tag{2.1.7}$$

Here, $X_{(tr)}$ is the weighted average value and it will be described in the next Section (2.1.4). $\|X_{(tr)} - x_{i(rec)}\|^2$ is the reconstruction error for the i^{th} sample and $\|f_i - W z_i\|^2$ is the linear regression loss for the i^{th} sample. α is a parameter that maintains a balance between both the terms in the decision metric. The value of α is empirically chosen to be 0.8.

2.1.4 Tracking Procedure

For the first frame of our video sequence, we use ground truth given for the first frame and annotate the target with a bounding box $x^1 = (px^1, py^1, h^1, w^1)$. Here, (px^1, py^1) is the centroid, h^1 and w^1 are the height and width of the bounding box annotated around the target respectively. Now, we randomly select N_0 positive and N_0 negative samples around x^1. Positive samples are obtained by randomly shifting the bounding box a few pixels around x^1, N_0 times. N_0 number of negative samples are obtained by randomly shifting the bounding box far away from x^1, N_0 times. In order to make sure that positive samples do not overlap with negative samples, the distance between the two is kept sufficiently large. The value of N_0 has been empirically fixed to 100. Now set X_p is created by combining the feature vectors obtained on applying histogram of gradients to the image patches around each of the N_0 positive samples. In a similar way X_n is found out for negative samples. Once we have

$X_0 = [X_p; X_n]$, we can obtain D_0 and W_0 as described in Section (2.1.2).

For the frames ahead, we randomly sample a fixed number of P particles around the centroid of the previous frame x^{t-1} based upon the Gaussian distribution $p(x^t/x^{t-1})$. Now we extract the feature vector x_i from a 32x32 image patch around the i^{th} particle, for each of the P particles. The feature vector x_i is extracted from the image patch by finding a 496 dimensional histogram of gradients feature using [21]. This is how we obtain a new data matrix $X = \{x_1, x_2, ..., x_P\}$. Then we compute sparse code matrix Z using Equation (2.1.3). Next, we compute the error vector $\varepsilon \in \mathbb{R}^{P\times 1}$ for all the sampled candidates using Equation (2.1.6). The particle with the smallest joint error value is chosen to be the best particle and chosen as the tracking result for the current frame. The value of P is chosen to be 800 empirically.

To calculate the reconstruction error term in Equation (2.1.7), we accumulate the feature vector extracted from the image patch around the best location into a set U. This set U always has a fixed number of elements U_{fix}. Whenever the number of elements in U exceeds U_{fix}, the elements from older frames are deleted. With each element in U, a weight $w = e^{-\varepsilon}$ is associated. Here, ε is the joint decision error for that element. X_{tr} in Equation (2.1.7) is computed as the weighted average of all the elements in U. This way the elements with smaller ε get more importance and the elements with larger ε get less importance on the combined sample X_{tr}. In the first frame U just contains one element which is the ground truth bounding box used to initialize the tracker. Thus, its weight is 1.

To run the update for D and W, another set V is constructed. In each frame after finding out the best particle x^t, we randomly sample N positive and N negative samples around x^t. Positive samples are obtained by randomly shifting the bounding box a few pixels around x^t, N times. N number of negative samples are obtained by randomly shifting the bounding box far away from x^t, N times. In order to make sure that positive samples do not overlap with negative samples, the distance between the two is kept sufficiently large. Now the feature vector for the positive samples (X_{pos}) is extracted using histogram of oriented gradients of the image patch around each of the positive samples. In a similar way X_{neg} is found out for negative samples. The distance between the negative and positive samples is kept sufficiently large, so that X_{neg} represents pure background. $X = [X_{pos}; X_{neg}]$ is added to set V. After accumulating X in V for a fixed V_{fix} number of frames, D is updated by solving Equation (2.1.1) for a fixed number of iterations. Then W is updated on Z obtained from Equation(2.1.1), using Equation (2.1.5). After the update, set V is emptied. The value of N has been empirically found out to be 100.

The tracking result obtained before might be noisy and may be inaccurate. Thus, we experimentally fix two thresholds th_1 and th_2. Before accumulating the tracking result and set $X = [X_{pos}; X_{neg}]$ for a particular frame into U and V respectively, we check if the reconstruction error as mentioned in Section (2.1.3) for the optimal particle for that frame is less than th_1 and likewise the regression loss is less than th_2. If this is true, we add the tracking result for that frame into U and $X = [X_{pos}; X_{neg}]$ into V. If any of the two errors is more than its respective threshold, we do not add the results and set X into sets U and V.

2.2 Multiple Analysis Dictionaries for tracking

Our algorithm using multiple analysis dictionaries is named as **OMADL**, Online Multiple Analysis Dictionary Learning.

2.2.1 Problem Formulation

We have two sets of input training samples X_1 and X_2. $X_1 = \{x_1^1, x_2^1, ..., x_n^1\}$ and $X_1 \in \mathbb{R}^{d_1 \times n}$. Here, $d_1 = 496$ is the dimension of HoG vector. $X_2 = \{x_1^2, x_2^2, ..., x_n^2\}$ and $X_2 \in \mathbb{R}^{d_2 \times n}$. Here, $d_2 = 1024$ is the dimension of LDP vector. Now our aim is to learn dictionary D_1 on HoG data samples and dictionary D_2 on LDP as given in [22] data samples. Both the dictionaries should solve a binary classification problem of discriminating foreground from the background. Therefore, the class labels are $Y_1 = \{1, -1\}$. Here, $label = 1$ corresponds to the target and $label = -1$ indicates the background. Any x_i^1 is a feature vector extracted from a 32x32 image patch using histogram of gradients for feature representation. Similarly, any x_i^2 is a feature vector extracted a 32x32 image patch using second order local derivative pattern. This image patch either corresponds to the positive sample which is our tracking target or a negative sample which is the background in a frame of a video sequence.

Given dictionaries $D_1 = \{d_1^1, d_2^1, ..., d_k^1\} \in \mathbb{R}^{k \times d_1}$ and $D_2 = \{d_1^2, d_2^2, ..., d_k^2\} \in \mathbb{R}^{k \times d_2}$. Here, k is the number of atoms in a dictionary. The sparse code corresponding to each x_i^1 is z_i^1 and each x_i^2 is z_i^2. In the analysis formulation each $z_i^1 \approx D_1 x_i^1$ and each $z_i^2 \approx D_2 x_i^2$. Thus, $z_i^1 \in \mathbb{R}^{k \times 1}$ and $z_i^2 \in \mathbb{R}^{k \times 1}$. The set of all sparse codes corresponding to HoG vectors combined together is $Z_1 = \{z_1^1, z_2^1, ..., z_n^1\}$. Similarly, the set of all sparse codes corresponding to LDP vectors combined together is $Z_2 = \{z_1^2, z_2^2, ..., z_n^2\}$ Thus, $Z_1 \approx D_1 X_1$ and $Z_1 \in \mathbb{R}^{k \times n}$. Likewise $Z_2 \approx D_2 X_2$ and $Z_2 \in \mathbb{R}^{k \times n}$ The loss function over D_1 and Z_1 is:

$$min_{D_1, Z_1} \|D_1 X_1 - Z_1\|_F^2 + \lambda_2 \|D_1\|_F^2 + \lambda_1 \|Z_1\|_1 \tag{2.2.1}$$

The loss function over D_2 and Z_2 is:

$$min_{D_2, Z_2} \|D_2 X_2 - Z_2\|_F^2 + \lambda_5 \|D_2\|_F^2 + \lambda_4 \|Z_2\|_1 \tag{2.2.2}$$

The term $\|D_1 X_1 - Z_1\|_F^2$ in Equation (2.2.1) is called the sparsification error for the data X_1 in the dictionary domain and the term $\|D_2 X_2 - Z_2\|_F^2$ in Equation (2.2.2) is the sparsification error for the data X_2 in the dictionary domain. Sparsification error is a modelling error in the dictionary representation. Our aim is to minimize this error in order to learn the best analysis dictionary model. Here λ_1, λ_2, λ_4 and λ_5 are regularization parameters.

After computing D_1 and Z_1, we find classifier W_1 on the value of Z_1 that has been calculated using Equation (2.2.1). Thus, we are solving two independent loss functions. One loss function to find D_1 and Z_1 and the other to find W_1. For formulating the loss function for calculating classifier W_1, we proceed as follows.

$$min_{W_1} \|F - W_1 Z_1\|_F^2 + \lambda_3 \|W_1\|_F^2 \tag{2.2.3}$$

The term $\|F - W_1 Z_1\|_F^2$ in the above equation is a linear regression loss or the classification error. Each column of F, which is f_i, is a label vector for x_i and $f_i \in \mathbb{R}^m$. So, $f_i = [1, 0]^T$ indicates a positive sample or class label$= 1$ and $f_i = [0, 1]^T$ indicates a negative sample or class label $= -1$. The classifier $W_1 \in \mathbb{R}^{m \times k}$. Number of rows in the classifier, $m = 2$, because there are two classes, one positive and the other negative.

Likewise, we find classifier W_2 on the value of Z_2 that has been calculated using Equation (2.2.2).

$$min_{W_2} \|F - W_2 Z_2\|_F^2 + \lambda_6 \|W_2\|_F^2 \tag{2.2.4}$$

Hence, for calculating second dictionary and classifier as well, we solve two independent loss functions.
The term $\|F - W_2 Z_2\|_F^2$ in the above equation is a linear regression loss. The classifier $W_2 \in \mathbb{R}^{m \times k}$. Number of rows in this classifier is also $m = 2$, because there are two classes, one positive and the other negative.

2.2.2 Initialization of HoG Dictionary and Classifier

KSVD[KSVD paper] is run on positive and negative HoG samples separately to form two dictionaries of the same size. These two are combined together to form a dictionary $D^1_{init} \in \mathbb{R}^{d_1 \times k}$. Now our initial HoG dictionary D^1_0 is the transpose of D^1_{init}. Thus, $D^1_0 = (D^1_{init})^T$ and $D^1_0 \in \mathbb{R}^{k \times d_1}$. The label f_i for each dictionary atom remains same in subsequent learning, only the value of each atom d^1_i is updated. Once we have D^1_0, the sparse code for HoG samples is calculated over the initial dictionary D_0^1:

$$Z_{calc}^1 = argmin_{Z_1} \|DX_1 - Z_1\|_F^2 + \lambda_1 \|Z_1\|_1 \tag{2.2.5}$$

Equation (2.2.5) is solved using soft thresholding with λ_1 as the threshold. Here, $D = D_0^1$. Now, once we have the initial dictionary D_0^1 and the sparse code Z_{calc}^1, we compute the initial classifier W_0^1. W_0^1 is computed using ridge regression,

$$W_0^1 = argmin_{W_1} \|F - W_1 Z_{calc}^1\|_F^2 + \lambda_3 \|W_1\|_F^2 \tag{2.2.6}$$

Here, F is the label matrix as mentioned in Section (2.1.1). The solution to Equation (2.2.6) is:

$$W_0^1 = F {Z_{calc}^1}^T (Z_{calc}^1 {Z_{calc}^1}^T + \lambda_3 I)^{-1} \tag{2.2.7}$$

Here, for any matrix A, A^T is the transpose of matrix A.

2.2.3 Initialization of LDP Dictionary and Classifier

KSVD [20] is run on positive and negative LDP samples separately to form two dictionaries of the same size. These two are combined together to form a dictionary $D_{init}^2 \in \mathbb{R}^{d_2 \times k}$. Now our initial LDP dictionary D_0^2 is the transpose of D_{init}^2. Thus, $D_0^2 = (D_{init}^2)^T$ and $D_0^2 \in \mathbb{R}^{k \times d_2}$. The label f_i for each LDP dictionary atom remains same in subsequent learning, only the value of each atom d_i^2 is updated. Once we have D_0^2, the sparse code for LDP samples is calculated over it.

$$Z_{calc}^2 = argmin_{Z_2} \|DX_2 - Z_2\|_F^2 + \lambda_4 \|Z_2\|_1 \tag{2.2.8}$$

Equation (2.2.8) is solved using soft thresholding with λ_4 as the thresholding parameter. Here, $D = D_0^2$. Now, once we have the initial dictionary D_0^2 and the sparse code Z_{calc}^2, we compute the initial classifier W_0^2. W_0^2 is computed using ridge regression,

$$W_0^2 = argmin_{W_2} \|F - W_2 Z_{calc}^2\|_F^2 + \lambda_6 \|W_2\|_F^2 \tag{2.2.9}$$

Here, F is the label matrix as mentioned in Section (2.1.1). The solution to Equation (2.2.9) is:

$$W_0^2 = F {Z_{calc}^2}^T (Z_{calc}^2 {Z_{calc}^2}^T + \lambda_6 I)^{-1} \tag{2.2.10}$$

Now we have both the initial dictionaries and both the initial classifier, we proceed as follows.

2.2.4 Classification

Once we have learned the dictionaries and classifiers, we classify new frames into foreground and background. For every new frame, P particles are sampled around the centroid of the previous frame. Now we extract the feature vector x_i^1 from a 32x32 image patch around the i^{th} particle, for each of the P particles. The feature vector x_i^1 is extracted from the image patch by finding a 496 dimensional histogram of gradients feature using [21]. This is how we obtain a new data matrix $X_1 = \{x_1^1, x_2^1, ...x_P^1\}$. We find the sparse code Z_1 of X_1 using Equation (2.2.5). Similarly, by finding second order local derivative pattern on the image patches, we get a new data matrix $X_2 = \{x_1^2, x_2^2, ...x_P^2\}$. We find the sparse code Z_2 of X_2 using Equation (2.2.5).

Now inspired by [9], we use a weighted joint decision measure for each sample during testing. For calculating this joint decision metric, we first find out reconstructed x_i^1 ($x_{i(rec)}^1$)

using the sparse code z_i^1 and dictionary D_1.

$$x_{i(rec)}^1 = (D_1^T D_1 + \lambda_1 I)^{-1} D_1^T z_i^1 \tag{2.2.11}$$

Likewise reconstructed x_i^2 is:

$$x_{i(rec)}^2 = (D_2^T D_2 + \lambda_4 I)^{-1} D_2^T z_i^2 \tag{2.2.12}$$

The weighted joint decision measure for HoG data samples is:

$$\varepsilon_1(x_i^1) = \|X_{tr}^1 - x_{i(rec)}^1\|^2 + \omega_i^1 \|f_i - W_1 z_i^1\|^2 \tag{2.2.13}$$

The weighted joint decision measure for LDP data samples is:

$$\varepsilon_2(x_i^2) = \|X_{tr}^2 - x_{i(rec)}^2\|^2 + \omega_i^2 \|f_i - W_2 z_i^2\|^2 \tag{2.2.14}$$

X_{tr}^1 in Equation (2.2.13) is the weighted average value of HoG features of the tracking result and X_{tr}^2 in Equation (2.2.14) is the weighted average value of LDP features of the tracking result. $\|X_{tr}^1 - x_{i(rec)}^1\|^2$ is the reconstruction error for the i^{th} HoG sample and $\|X_{tr}^2 - x_{i(rec)}^2\|^2$ is the reconstruction error for the i^{th} LDP sample. $\|f_i - W z_i^1\|^2$ is the linear regression loss for the i^{th} HoG sample and $\|f_i - W z_i^2\|^2$ is the linear regression loss for the i^{th} LDP sample. We add a weight to classification error, so that its contribution to the error metric gets updated dynamically. If a feature describes the target with higher ability, then the weight of classification error associated to that feature would be high, else small.

Calculating Weights

To calculate the decision metrics in Equation (2.2.13) and Equation (2.2.14), we find out the weighting parameters ω_i^1 and ω_i^2 as calculated in [9].

$$\omega_i^1 = \rho(1 - \frac{\|X_{tr}^1 - x_{i(rec)}^1\|^2}{\|X_{tr}^1 - x_{i(rec)}^1\|^2 + \|X_{tr}^2 - x_{i(rec)}^2\|^2}) \tag{2.2.15}$$

$$\omega_i^2 = \rho(1 - \frac{\|X_{tr}^2 - x_{i(rec)}^2\|^2}{\|X_{tr}^1 - x_{i(rec)}^1\|^2 + \|X_{tr}^2 - x_{i(rec)}^2\|^2}) \tag{2.2.16}$$

Here, ρ is a constant to control the contribution of weight to classification error and has been fixed to 0.5 experimentally.

The weight attached to a feature is a measure of its dictionary's capability to describe the target. Thus, to calculate the weights we use the reconstruction error term described before in the same Section. This way we make our appearance model more robust to various situations and challenges. The terms X_{tr}^1 and X_{tr}^2 are explained in the next section.

2.2.5 Tracking Procedure

For the first frame of our video sequence, we use ground truth given for the first frame and annotate the target with a bounding box $x^1 = (px^1, py^1, h^1, w^1)$. Here, (px^1, py^1) is the centroid, h^1 and w^1 are the height and width of the bounding box annotated around the target respectively. Now, we randomly select N_0 positive and N_0 negative samples around x^1. N_0 Positive samples are obtained by randomly shifting the bounding box a few pixels around x^1. Negative samples are obtained by randomly shifting the bounding box far away from x^1 N_0 times. In order to make sure that positive samples do not overlap with negative samples, the distance between the two is kept sufficiently large. The value of N_0 has been empirically fixed to 100. Now set (X_p^1) is created by combining the feature vectors obtained on applying histogram of gradients to the image patches around each of the N_0 positive samples. In a similar way X_n^1 is found out for negative samples. Once we have $X_0^1 = [X_p^1; X_n^1]$, we can obtain D_0^1 and W_0^1 as described in Section (2.2.2). Similarly, set (X_p^2) is created by combining the feature vectors obtained on applying second order local derivative pattern to the image patches around each of the N_0 positive samples. In a similar way X_n^2 is found out for negative samples. Once we have $X_0^2 = [X_p^2; X_n^2]$, we can obtain D_0^2 and W_0^2 as described in Section (2.2.3).

For subsequent frames, we randomly sample a fixed number of P particles around the centroid of the previous frame x^{t-1} based upon the Gaussian distribution $p(x^t/x^{t-1})$. Now we extract HoG feature vector $x_i{}^1$ and LDP feature vector $x_i{}^2$ from a 32x32 image patch around the i^{th} particle, for each of the P particles. This is how we obtain new data matrices $X_1 = \{x_1^1, x_2^1, ..., x_P^1\}$ and $X_2 = \{x_1^2, x_2^2, ..., x_P^2\}$. Then we compute sparse code matrices Z_1 and Z_2 using Equation (2.2.5) and Equation (2.2.6) respectively.

Next, we compute an error vector $\varepsilon_1 \in \mathbb{R}^{P\times 1}$, where the i^{th} element is the joint decision metric in Equation (2.2.13) for the i^{th} candidate. In the same way error vector $\varepsilon_2 \in \mathbb{R}^{P\times 1}$ is calculated, where the i^{th} element is the joint decision metric in Equation (2.2.14) for the i^{th} candidate. Now a total error vector ε is calculated by:

$$\varepsilon = \varepsilon_1 + \varepsilon_2 \tag{2.2.17}$$

The particle with the smallest joint error value is chosen to be the best particle and chosen as the tracking result for the current frame. The value of P is chosen to be 800 empirically.

Now to calculate X_{tr}^1 for finding the reconstruction error term in Equation (2.2.13), we accumulate the HoG feature vector extracted from the image patch around the best location into a set U_1. This set U_1 always has a fixed number of elements U_{fix}^1. Whenever the number of elements in U_1 exceeds U_{fix}^1, the element from the oldest frame is deleted. With each element in U_1, a weight $e^{-\varepsilon_1}$ is associated. Here, ε_1 is the joint decision error for that element calculated using Equation (2.2.13). X_{tr}^1 is computed as the weighted average of all the elements in U_1. This way the elements with smaller ε_1 get more importance and the elements with larger ε_1 get less importance on the combined sample X^1_{tr}. In the first frame U_1 just contains one element which is the ground truth bounding box used to initialize the tracker. Thus, its weight is initialized to 1.

Likewise, we calculate X^2_{tr} for finding the reconstruction error term in Equation (2.2.14), we accumulate the LDP feature vector extracted from the image patch around the best location into a set U_2. The set U_2 always has a fixed number of elements U^2_{fix}. As soon as the number of elements in U_2 exceeds U^2_{fix}, element from the oldest frame is deleted. A weight $e^{-\varepsilon_2}$ is associated with each element in U_2. Here, ε_2 is the joint decision error for that element calculated using Equation (2.2.14). X^2_{tr} is the weighted average of all the elements in U_2. In the first frame U_2 just contains one element which is the ground truth bounding box used to initialize the tracker and its weight is initialized to 1.

To run the update for D_1 and W_1, another set V_1 is constructed. In each frame after finding out the best particle x^t, we randomly sample N positive and N negative samples around x^t. Positive samples are obtained by randomly shifting the bounding box a few pixels around x^t, N times. N number of negative samples are obtained by randomly shifting the bounding box far away from x^t, N times. In order to make sure that positive samples do not overlap with negative samples, the distance between the two is kept sufficiently large. Now set X^1_{pos} is created by combining the feature vectors obtained on applying histogram of gradients to the image patches around each of the N positive samples. In a similar way X^1_{neg} is found out for negative samples. The distance between the negative and positive samples is kept sufficiently large, so that X^1_{neg} represents pure background. $X_1 = [X^1_{pos}; X^1_{neg}]$ is added to set V_1. We keep accumulating X_1 in V_1 for a fixed V^1_{fix} number of frames and then D_1 is updated by solving Equation (2.2.1) for a fixed number of iterations. Then W_2 is updated on Z_1 obtained from Equation(2.2.5). After the update, set V_1 is emptied.

Similarly, for updating D_2 and W_2, we construct another set V_2. In each frame after finding out the best particle x^t, we randomly sample N positive and N negative samples around x^t as explained above in the same Section. Now set X_{pos}^2 is constructed by combining the feature vectors obtained on applying second order local derivative pattern to the image patches around each of the N positive samples. X_{neg}^2 is found out for negative samples by the same process. The distance between the negative and positive samples is kept sufficiently large, so that X_{neg}^2 represents only background. $X_2 = [X_{pos}^2; X_{neg}^2]$ is added to set V_2. We keep accumulating X_2 in V_2 for a fixed V_{fix}^2 number of frames and then D_2 is updated by solving Equation (2.2.2) for a fixed number of iterations. Then W_2 is updated on Z_2 obtained from Equation(2.2.8). After the update, set V_2 is emptied. The value of N has been empirically found out to be 100.

The tracking result obtained before might be noisy and may not accurately represent the target. Thus, we have fixed two pairs of thresholds th_1^1, th_2^1 and th_1^2, th_2^2 empirically. Before accumulating the HoG feature vector around the best particle and set $X_1 = [X_{pos}^1; X_{neg}^1]$ for a particular frame into U_1 and V_1 respectively, we check if the reconstruction error of HoG dictionary as mentioned in Section (2.2.4) for the best particle of that frame is less than th_1^1 and the regression loss is less than th_2^1. If this is true, we add the HoG feature vector around the best particle for that frame into U_1 and $X_1 = [X_{pos}^1; X_{neg}^1]$ into V_1. If any of the two errors is more than its respective threshold, we do not add the HoG vector and set X_1 into sets U_1 and V_1 respectively.

Similarly, before we accumulate the LDP feature vector around the best particle and set $X_2 = [X_{pos}^2; X_{neg}^2]$ for a particular frame into U_2 and V_2 respectively, we check if the reconstruction error of LDP dictionary as mentioned in Section (2.2.4) for the best particle of that frame is less than th_1^2 and the regression loss is less than th_2^2. If this is true, we add the LDP feature vector around the best particle for that frame into U_2 and $X_2 = [X_{pos}^2; X_{neg}^2]$ into V_2. If any of the two errors is more than its respective threshold, we do not add the LDP feature vector and set X_2 into U_2 and V_2 respectively.

Chapter 3

Experimental Settings and Results

This chapter describes all the experimental settings and simulations for evaluating the performance of both the trackers. There are 50 videos in OTB-50 dataset in [23] which cover all the eleven attributes (illumination variation, occlusion, deformation etc.) that affect a tracker's performance. The two proposed trackers are made to run on this dataset and their overall performance as well as performance against these eleven attributes individually is assessed.

The metrics for evaluating their performance are: Precision Plots and Success Plots. The underlying metrics to generate these plots are Central Location Error and Intersection Over Union Score respectively. This section briefly explains both the metrics and the methodology of generating the plots using these metrics. The performance of our trackers using these plots is compared with all the trackers mentioned in Visual Tracking Benchmark [23] and some other state of the art trackers like [1, 12, 13, 15–17].

This section also gives a detailed account of all the parameters and their values that were chosen for running the experiments. The two proposed trackers were coded in MATLAB and all the experiments were performed on MATLAB R2014a running on a 64 bit Windows machine having 4 GB RAM and Intel i5 CPU @ 2.5 GHz.

3.1 Evaluation Methodology

As in [23], Success Plots and Precision Plots are used for quantitative assessment of both the proposed trackers. The two plots are generated as under:

3.1.1 Precision Plots

Center location Error is an error metric used for evaluating the performance of tracking algorithms. It is defined as the Euclidean distance between the centres of the bounding box predicted by the tracker and the manually labelled ground truth. Then the average center location error over all the frames of one sequence is calculated. This average center location error gives the tracking performance for that sequence.

The average center location is not sufficient alone to evaluate the performance of trackers because when the tracker will lose the target, the predicted bounding box location can be random and the average error value may not accurately measure the tracking performance. It was argued by [24] that precision plot is a better metric to evaluate trackers than centre location error.

The precision plot is generated in the following way. A range of values of pixel distances from 0 to 50 units is chosen as the threshold. The percentage of frames in which the estimated location of the centre of bounding box is within a particular value of threshold distance when compared to ground truth is plotted against the threshold values to generate the precision plots. These plots are simple and easy to understand. As in [16] more accurate trackers have high values of precision at lower thresholds values. If a tracker is not able to reach precision values equal to 1 for a large range of threshold, it indicates its performance is poor.

To calculate the ranking of the trackers based on precision plots, a threshold value of 20 pixels is chosen as explained in [23]. This is how precision plots are produced and ranking of trackers based on precision plots is done.

3.1.2 Success Plots

Intersection over unions is another common metric frequently used for evaluating the performance of trackers. Given the resultant bounding box R_T , as predicted by the proposed tracker for a frame and the ground truth bounding box R_{GT}, then the intersection over union metric is calculated by $IOU = \frac{area(R_{GT} \cap R_T)}{area(R_{GT} \cup R_T)}$. Here, $\cap$ is intersection and $\cup$ is union of two regions.

Earlier, a specific threshold used to be selected(for eg. 0.5) and all frames in which the IOU was above this threshold were treated as successful. But this method is not very appropriate to quantify the performance of trackers. Thus, [23] introduced success plots to measure the performance of trackers.

To generate the success plots, the threshold is varied from 0 to 1. The number of frames in which the IOU is greater than the given threshold, are considered as successful for that value of threshold. The ratio of successful frames to total frames is plot against threshold values. This is how the success plots are generated.

For the ranking of trackers based on success plots, the Area Under Curve(AUC) is calculated for the success plot of each tracking algorithm. The trackers are ranked on the basis of decreasing order of these AUC values.

3.2 Feature Descriptors

In our implementation using single analysis dictionary (OADL), we use a 496-dimensional histograms of oriented gradients (HoG) feature. The objects that are being tracked are mostly human and animal faces, humans, vehicles etc. For discriminating these objects from the background, using the information encoded in edges and corners is very useful because edges and corners convey a lot of information about object shape. There is a sharp change in intensity around edges and corners and the magnitude of gradients is considerably large around them. Thus, using a descriptor in which the underlying features are gradients is a suitable choice for classification task.

In our implementation using two analysis dictionaries (OMADL), we use a 496-dimensional histograms of oriented gradients (HoG) feature [19] and second order local derivative pattern(LDP) with 1024-dimensional LDP feature vector to generate the dictionaries. To model the distribution of second order LDP we have used a histogram as in [22]. The reason for using HoG has already been explained. The second-order LDP captures the change of derivative directions in the local neighbourhood, and encodes the change in a given direction. More extensive discriminative information can be captured by a second order descriptor in a frame. Thus, LDP in conjunction with HoG gives a considerably good performance.

3.3 Parameter Selection

3.3.1 Parameters for Single Analysis Dictionary Model

The values of parameters used for single analysis dictionaries are in the table below :

Table 3.3.1: Parameters in Single Analysis Dictionary model

Parameter	**Value**
λ_1	0.01
λ_2	0.00001
λ_3	1000
No. of dictionary atoms	200
No. of optimal candidates in set U	20
No. of frames after set V is emptied	4

In the above table, λ_1, λ_2 and λ_3 are regularization parameters as explained before. The number of dictionary atoms were chosen empirically. Lesser number of atoms deteriorated the performance of tracking algorithm and choosing more number of atoms decreased its speed. Thus, experimentally 200 atoms was a good trade-off between the speed and performance of the tracker. Out of 200 dictionary atoms, 100 atoms represent positive samples and 100 atoms represent negative samples. The number of iterations for dictionary learning during initialization are 5. The number of iterations have also been fixed experimentally. The number of optimal location candidates in U, that is the number of optimal location candidates stored from the previous frames is 20. As soon as the number of candidates become more than 20, the candidates from oldest frame are deleted. Thus, U always has a fixed number of elements stored in it. The number of frames after which the dictionary and classifier are updated or set V is emptied are 4. The threshold value th_1 for reconstruction error is 0.35 and the the threshold value th_2 for regression loss or classification error is 1. If any of the two errors are more than their respective threshold values, the tracking result of that frame is discarded and not saved in set U. The value of parameter α that maintains a trade off between the classification error and linear regression loss is 0.8. All of these have been found out empirically and their values remain same for all the videos sequences in the dataset.

3.3.2 Parameters for Multiple Analysis Dictionaries Model

In the table for multiple analysis dictionaries model (OMADL) given on the next page, λ_1, λ_2 and λ_3 are regularization parameters for the dictionary with HoG features as explained before. λ_4, λ_5 and λ_6 are regularization parameters for the dictionary with LDP features. The number of dictionary atoms are 200 for both the dictionaries. The number of atoms representing positive and negative samples are 100 each. The number of iterations for learning the HoG dictionary during initialization are 5. The number of iterations for learning the LDP dictionary during initialization are 5. The number of optimal location candidates in U, that is the number of optimal location candidates stored from the previous frames is 20. As soon as the number of candidates become more than 20, the candidates from oldest frame are deleted.

Table 3.3.2: Parameters used in Multiple Analysis Dictionaries Model

Parameter	**Value**
λ_1	0.01
λ_2	0.00001
λ_3	1000
λ_4	0.001
λ_5	100
λ_6	10
No. of dictionary atoms	200
No. of optimal candidates in set U_1 and U_2	20
No. of frames after which sets V_1 and V_2 are updated	4

The number of frames after which the both the dictionaries and classifiers are updated or sets V_1 and V_2 are emptied are 4. The threshold value th_1 for reconstruction error is 0.35 and the threshold value th_2 for regression loss or classification error is 1 for both HoG and LDP dictionaries. If any of the two errors are more than their respective threshold values for any of the two dictionaries, the tracking result of that frame is discarded and not saved in set U. The value of parameter ρ that controls the contribution of the weight of each dictionary in the error metric is 0.5 for both the dictionaries. All of these have been found out empirically and their values remain same for all the videos sequences in the dataset.

3.4 Results

We run both the proposed algorithms on all the video sequences in OTB-50 dataset. We used the results of all the publically available trackers[2, 3, 7, 8, 17, 18, 24–44] computed on OTB-50 dataset as provided by [23] and the codes of MDNET [12], MEEM [13], DSST [14], KCF [16], TGPR [15] and ODDL [1] have been made available by the authors online. They were downloaded and were run on the OTB-50 dataset on our machine.

The performance of trackers has been shown by Success and Precision Plots and the tracked results of the trackers on frames of some video sequences from the dataset. Our proposed algorithm using single analysis dictionary is Online Analysis Dictionary Learning **OADL**. Our proposed algorithm using multiple analysis dictionary is Online Multiple Analysis Dictionary Learning **OMADL**.

3.4.1 Plots and Tables

The overall success and precision plots computed on all the video sequences in the dataset and the success and precision plots for all the eleven conditions include top 10 trackers in each case. For each plot we have computed a table that compares the Area Under Curve for each tracker and shows ranking of trackers based on both the plots. The ranking of trackers in success and precision plots has been done as described in section 3.1. Each table includes top 10 trackers along with the ranking and AUC for ODDL because our work is inspired by [1].

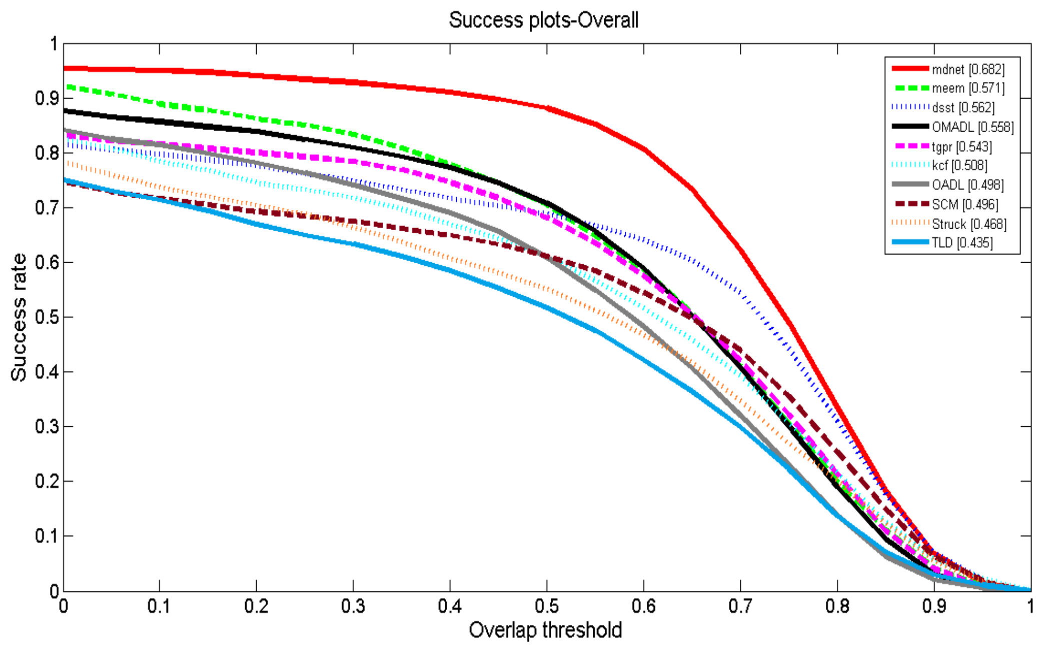


Figure 3.4.1: Overall-Success Plot

Table 3.4.1: Success Plots-Overall

Tracker	**AUC(Area Under Curve)**	**Rank**
MDNET	0.682	1
MEEM	0.571	2
DSST	0.562	3
OMADL	0.558	4
TGPR	0.543	5
KCF	0.508	6
OADL	0.498	7
SCM	0.496	8
STRUCK	0.468	9
TLD	0.435	10
ODDL	0.380	18

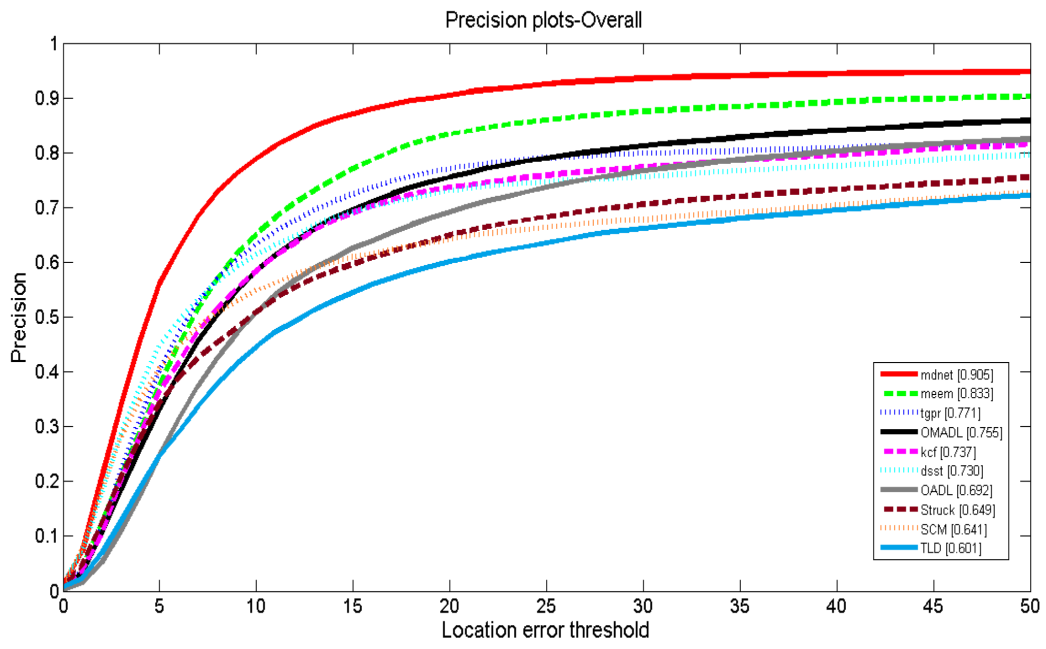


Figure 3.4.2: The overall precision plot over all the video sequences in OTB-50 dataset.

Table 3.4.2: Precision Plots-Overall

Tracker	**CLE = 20 pixels**	**Rank**
MDNET	0.905	1
MEEM	0.833	2
TGPR	0.771	3
OMADL	0.755	4
KCF	0.737	5
DSST	0.730	6
OADL	0.692	7
STRUCK	0.649	8
SCM	0.641	9
TLD	0.601	10
ODDL	0.525	14

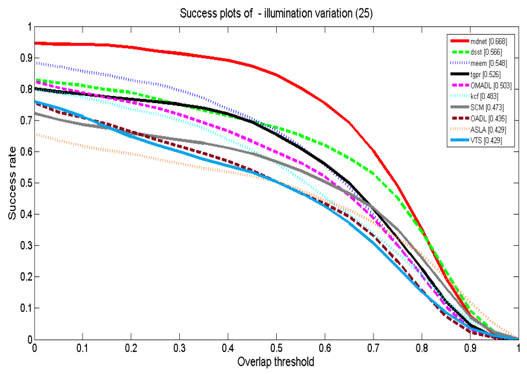


Figure 3.4.3: Success plot-Illumination Variation

Tracker	**AUC(Area Under Curve)**	**Rank**
MDNET	0.668	1
DSST	0.566	2
MEEM	0.548	3
TGPR	0.526	4
OMADL	0.503	5
KCF	0.483	6
SCM	0.473	7
OADL	0.435	8
ASLA	0.429	9
VTS	0.429	9
STRUCK	0.428	10
ODDL	0.348	18

Table 3.4.3: Success Plots-Illumination Variation

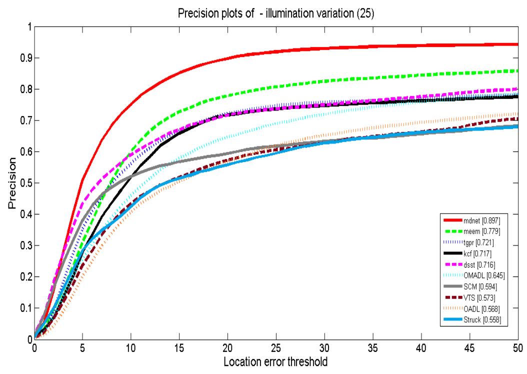


Figure 3.4.4: Precision plot-Illumination Variation

Tracker	**CLE = 20 pixels**	**Rank**
MDNET	0.897	1
MEEM	0.779	2
TGPR	0.721	3
KCF	0.717	4
DSST	0.716	5
OMADL	0.645	6
SCM	0.594	7
VTS	0.573	8
OADL	0.568	9
STRUCK	0.558	10
ODDL	0.459	17

Table 3.4.4: Precision Plots-Illumination Variation

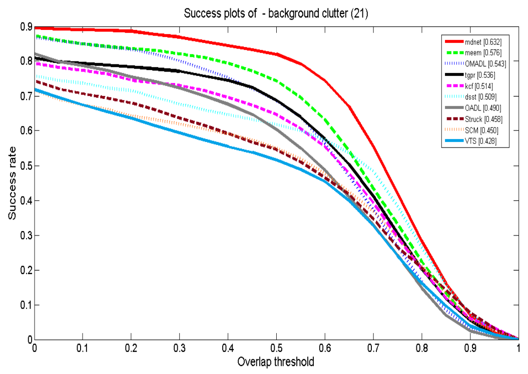


Figure 3.4.5: Success plot-Background Clutter

Tracker	AUC(Area Under Curve)	Rank
MDNET	0.632	1
MEEM	0.576	2
OMADL	0.543	3
TGPR	0.536	4
KCF	0.514	5
DSST	0.509	6
OADL	0.490	7
STRUCK	0.458	8
SCM	0.450	9
VTS	0.428	10
ODDL	0.363	18

Table 3.4.5: Success Plots- Background Clutter

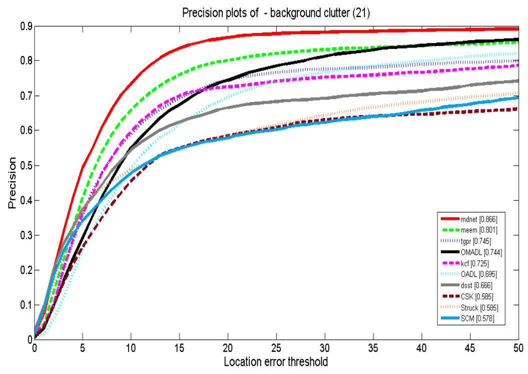


Figure 3.4.6: Precision Plot - Background Clutter

Tracker	CLE = 20 pixels	Rank
MDNET	0.866	1
MEEM	0.801	2
TGPR	0.745	3
OMADL	0.744	4
KCF	0.725	5
OADL	0.695	6
DSST	0.666	7
CSK	0.585	8
STRUCK	0.585	8
SCM	0.578	9
VTS	0.578	9
ODDL	0.529	12

Table 3.4.6: Precision Plots- Background Clutter

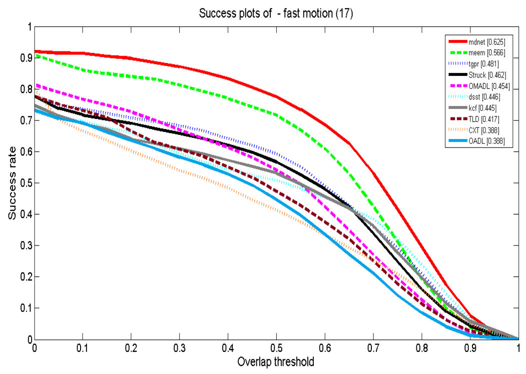


Figure 3.4.7: Success plot-Fast Motion

Tracker	AUC(Area Under Curve)	Rank
MDNET	0.625	1
MEEM	0.566	2
TGPR	0.481	3
STRUCK	0.462	4
OMADL	0.454	5
DSST	0.446	6
KCF	0.445	7
TLD	0.417	8
CXT	0.388	9
OADL	0.388	9
OAB	0.358	10
ODDL	0.291	25

Table 3.4.7: Success Plots-Fast Motion

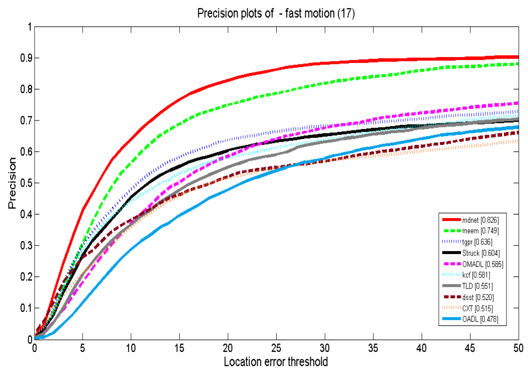


Figure 3.4.8: Precision Plot-Fast Motion

Tracker	CLE = 20 pixels	Rank
MDNET	0.826	1
MEEM	0.749	2
TGPR	0.636	3
STRUCK	0.604	4
OMADL	0.585	5
KCF	0.581	6
TLD	0.551	7
DSST	0.520	8
CXT	0.515	9
OADL	0.478	10
ODDL	0.341	26

Table 3.4.8: Precision Plot-Fast Motion

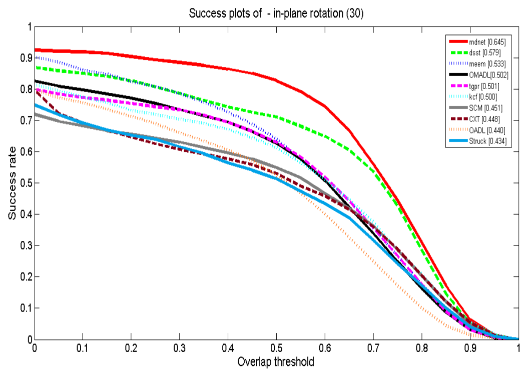


Figure 3.4.9: Success plot-In-plane Rotation

Tracker	AUC(Area Under Curve)	Rank
MDNET	0.645	1
DSST	0.579	2
MEEM	0.533	3
OMADL	0.502	4
TGPR	0.501	5
KCF	0.500	6
SCM	0.451	7
CXT	0.448	8
OADL	0.440	9
STRUCK	0.4343	10
ODDL	0.373	19

Table 3.4.9: Success Plots- In-plane Rotation

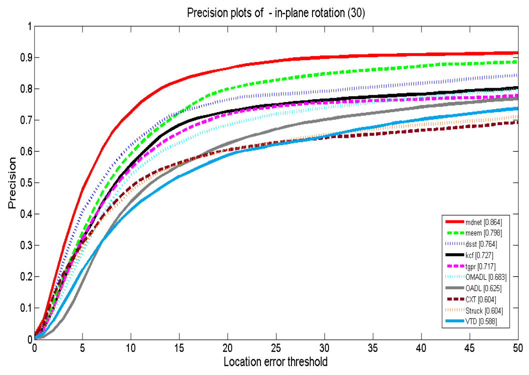


Figure 3.4.10: Precision plot-in plane rotation

Tracker	CLE = 20 pixels	Rank
MDNET	0.864	1
MEEM	0.798	2
DSST	0.764	3
KCF	0.727	4
TGPR	0.717	5
OMADL	0.683	6
OADL	0.625	7
CXT	0.604	8
STRUCK	0.604	8
VTD	0.588	9
SCM	0.583	10
ODDL	0.508	16

Table 3.4.10: Precision Plots- In-plane Rotation

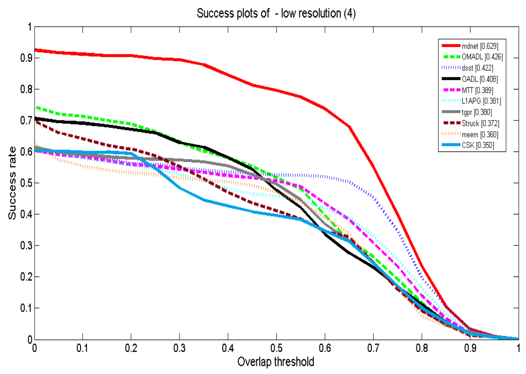


Figure 3.4.11: Success plot-Low Resolution

Tracker	CLE = 20 pixels	Rank
MDNET	0.629	1
OMADL	0.426	2
DSST	0.422	3
OADL	0.402	4
MTT	0.389	5
L1APG	0.381	6
TGPR	0.380	7
STRUCK	0.372	8
MEEM	0.360	9
CSK	0.350	9
SemiT	0.330	10
ODDL	0.293	16

Table 3.4.11: Success Plots-Low Resolution

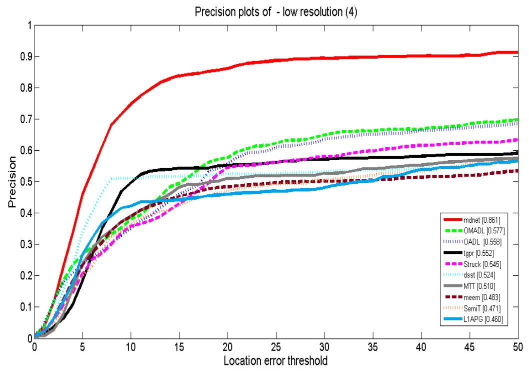


Figure 3.4.12: Precision plot- Low Resolution

Tracker	CLE = 20 pixels	Rank
MDNET	0.861	1
OMADL	0.577	2
OADL	0.558	3
TGPR	0.552	4
STRUCK	0.545	5
DSST	0.524	6
MTT	0.510	7
MEEM	0.483	8
SemiT	0.471	9
L1APG	0.460	10
ODDL	0.393	12

Table 3.4.12: Precision Plots-Low Resolution

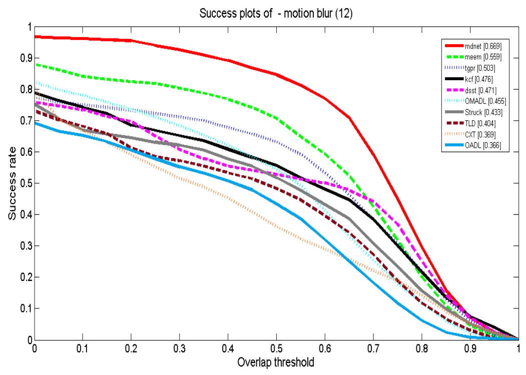


Figure 3.4.13: Success plot-Motion Blur

Tracker	AUC(Area Under Curve)	Rank
MDNET	0.669	1
MEEM	0.559	2
TGPR	0.503	3
KCF	0.476	4
DSST	0.471	5
OMADL	0.455	6
STRUCK	0.433	7
TLD	0.404	8
CXT	0.369	9
OADL	0.366	10
ODDL	0.290	24

Table 3.4.13: Success Plots- Motion Blur

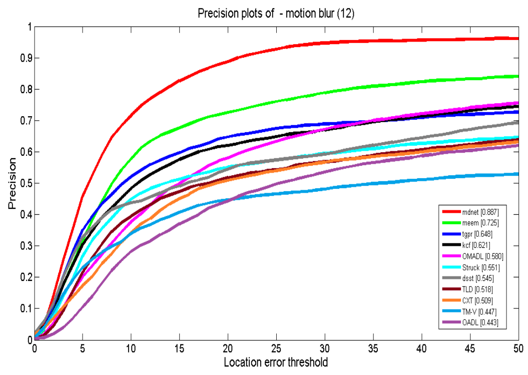


Figure 3.4.14: Precision plot-Motion Blur

Tracker	CLE = 20 pixels	Rank
MDNET	0.887	1
MEEM	0.725	2
TGPR	0.648	3
KCF	0.621	4
OMADL	0.580	5
STRUCK	0.551	6
DSST	0.545	7
TLD	0.518	8
CXT	0.509	9
TM-V	0.447	10
OADL	0.443	11
ODDL	0.341	20

Table 3.4.14: Precision Plots- Motion Blur

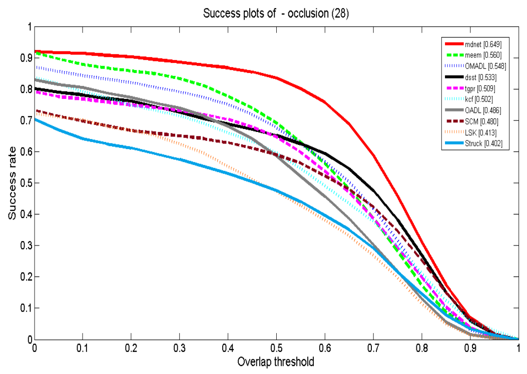


Figure 3.4.15: Success plots-Occlusion

Tracker	**AUC(Area Under Curve)**	**Rank**
MDNET	0.649	1
MEEM	0.560	2
OMADL	0.548	3
DSST	0.533	4
TGPR	0.509	5
KCF	0.502	6
OADL	0.486	7
SCM	0.480	8
LSK	0.413	9
STRUCK	0.402	10
ODDL	0.349	20

Table 3.4.15: Success Plots- Occlusion

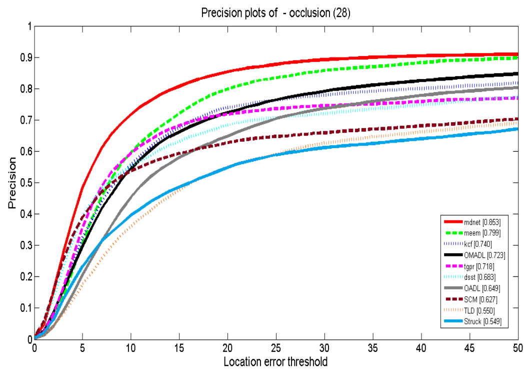


Figure 3.4.16: Precision plot- Occlusion

Tracker	**CLE = 20 pixels**	**Rank**
MDNET	0.853	1
MEEM	0.799	2
KCF	0.740	3
OMADL	0.723	4
TGPR	0.718	5
DSST	0.683	6
OADL	0.649	7
SCM	0.627	8
TLD	0.550	9
STRUCK	0.549	10
ODDL	0.458	21

Table 3.4.16: Precision Plots- Occlusion

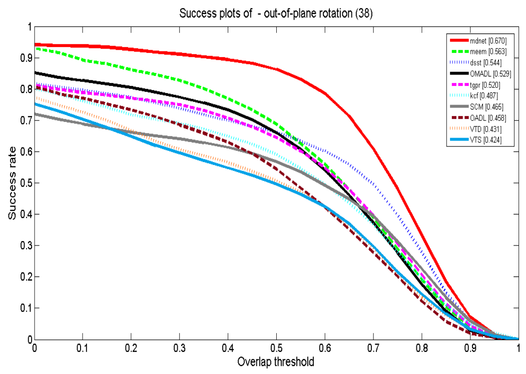


Figure 3.4.17: Success plots-Out-Of Plane Rotation

Tracker	AUC(Area Under Curve)	Rank
MDNET	0.670	1
MEEM	0.563	2
DSST	0.544	3
OMADL	0.529	4
TGPR	0.520	5
KCF	0.487	6
SCM	0.465	7
OADL	0.458	8
VTD	0.431	9
VTS	0.424	10
STRUCK	0.424	10
ODDL	0.356	19

Table 3.4.17: Success Plots- Out Of Plane Rotation

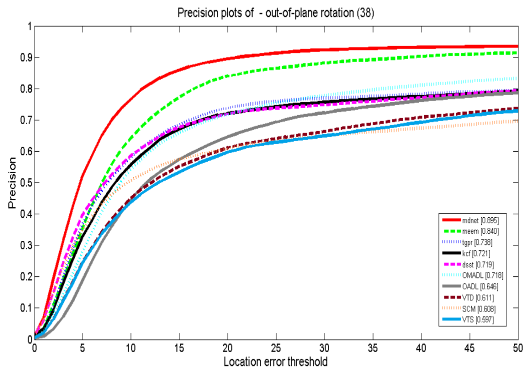


Figure 3.4.18: Precision plot - Out of plane Rotation

Tracker	CLE = 20 pixels	Rank
MDNET	0.895	1
MEEM	0.840	2
TGPR	0.738	3
KCF	0.721	4
DSST	0.719	5
OMADL	0.718	6
OADL	0.646	7
VTD	0.611	8
SCM	0.608	9
VTS	0.597	10
ODDL	0.486	22

Table 3.4.18: Precision Plots- Out Of Plane Rotation

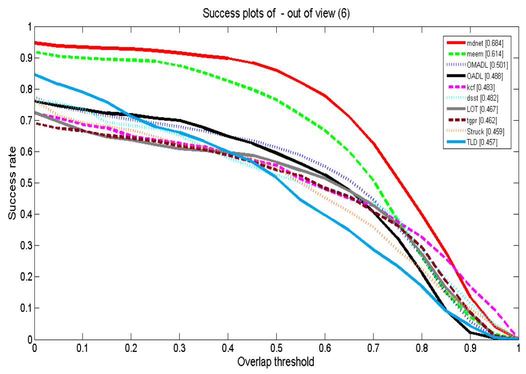


Figure 3.4.19: Success plots-Out Of View

Tracker	AUC(Area Under Curve)	Rank
MDNET	0.684	1
MEEM	0.614	2
OMADL	0.501	3
OADL	0.488	4
KCF	0.483	5
DSST	0.482	6
LOT	0.467	7
TGPR	0.462	8
STRUCK	0.459	9
TLD	0.457	10
ODDL	0.371	20

Table 3.4.19: Success Plots-Out Of View

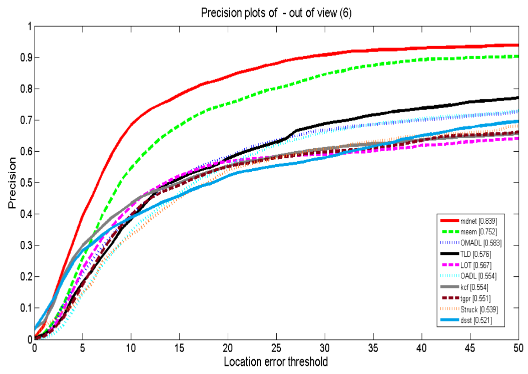


Figure 3.4.20: Precision plot- Out of View

Tracker	CLE = 20 pixels	Rank
MDNET	0.839	1
MEEM	0.752	2
OMADL	0.583	3
TLD	0.576	4
LOT	0.567	5
OADL	0.554	6
KCF	0.554	6
TGPR	0.551	7
DSST	0.521	8
LSK	0.515	9
CXT	0.510	10
ODDL	0.436	17

Table 3.4.20: Precision Plots-Out Of View

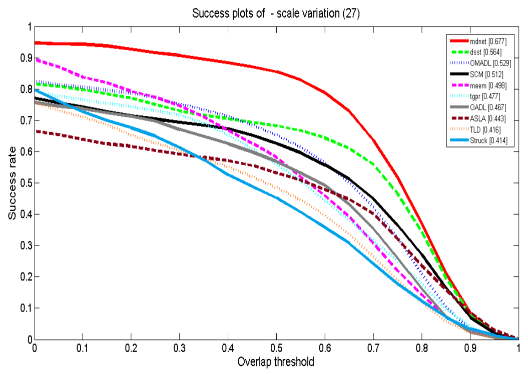


Figure 3.4.21: Success plots- Scale Variation

Tracker	**AUC(Area Under Curve)**	**Rank**
MDNET	0.677	1
DSST	0.564	2
OMADL	0.529	3
SCM	0.512	4
MEEM	0.498	5
TGPR	0.477	6
OADL	0.467	7
ASLA	0.443	8
TLD	0.416	9
STRUCK	0.414	10
ODDL	0.377	19

Table 3.4.21: Success Plots- Scale Variation

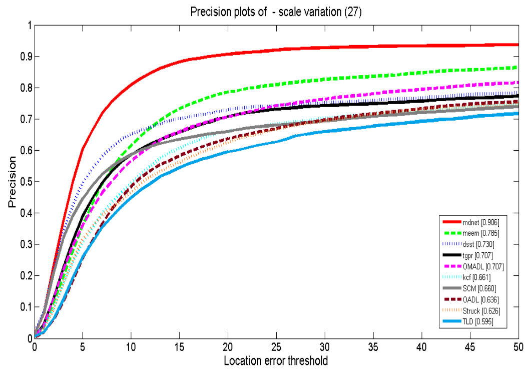


Figure 3.4.22: Precision plot -Scale Variation

Tracker	**CLE = 20 pixels**	**Rank**
MDNET	0.906	1
MEEM	0.785	2
DSST	0.730	3
TGPR	0.707	4
OMADL	0.707	4
KCF	0.661	5
SCM	0.660	6
OADL	0.636	7
STRUCK	0.626	8
TLD	0.595	9
VTD	0.584	10
ODDL	0.467	18

Table 3.4.22: Precision Plots- Scale Variation

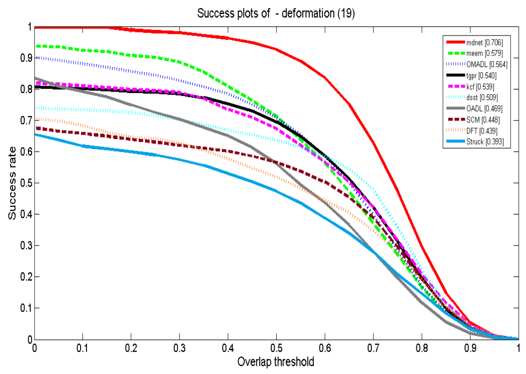


Figure 3.4.23: Success plot-Deformation

Tracker	**AUC(Area Under Curve)**	**Rank**
MDNET	0.706	1
MEEM	0.579	2
OMADL	0.564	3
TGPR	0.540	4
KCF	0.539	5
DSST	0.509	6
OADL	0.469	7
SCM	0.448	8
DFT	0.439	9
STRUCK	0.393	10
ODDL	0.354	20

Table 3.4.23: Success Plots- Deformation

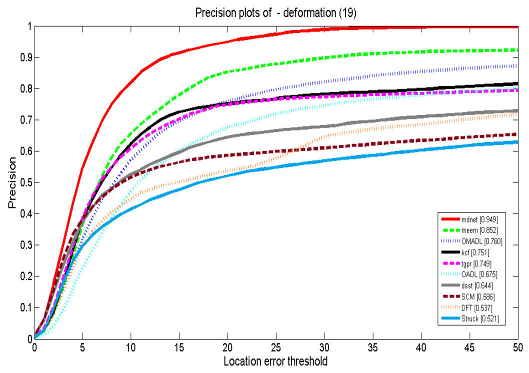


Figure 3.4.24: Precision plot-Deformation

Tracker	**CLE = 20 pixels**	**Rank**
MDNET	0.949	1
MEEM	0.852	2
OMADL	0.760	3
KCF	0.751	4
TGPR	0.749	5
OADL	0.675	6
DSST	0.644	7
SCM	0.586	8
DFT	0.537	9
STRUCK	0.521	10
ODDL	0.486	15

Table 3.4.24: Precision Plots- Deformation

Chapter 4

Discussion

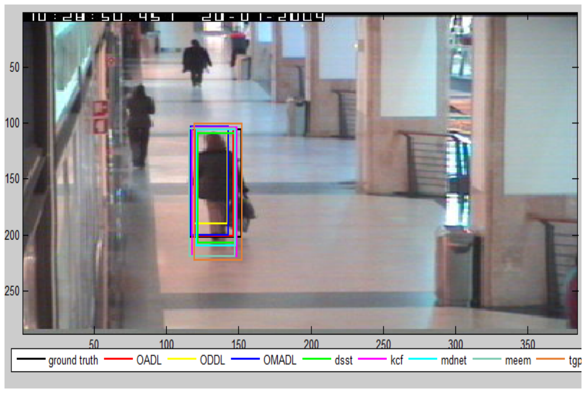
4.1 Quantitative Comparison with other trackers

As can be seen from the Success and Precision plots above, our approach **OADL** is among top 10 trackers for all plots exceept for precision plot of Motion blur, where it ranks 11^{th}. Clearly, our approach **OMADL** is an improvement over our approach using **OADL**. OMADL is among top 5 trackers for all the situations except in precision plots it stands 6^{th} for illumination variations, in-plane-rotations and out-of-Plane Rotations. In success plots it ranks 6^{th} in motion blur. The approach using multiple analysis dictionaries works fairly well for most of the situations because the analysis dictionaries that are learnt are able to classify the object from its background accurately. This is because the dictionaries are learnt on HoG and LDP features. HoG captures first order derivatives and LDP captures second order derivatives. Hence, both are non overlapping features and help to learn dictionaries which when used jointly with a weighting measure, have a good ability to distinguish foreground from the background.

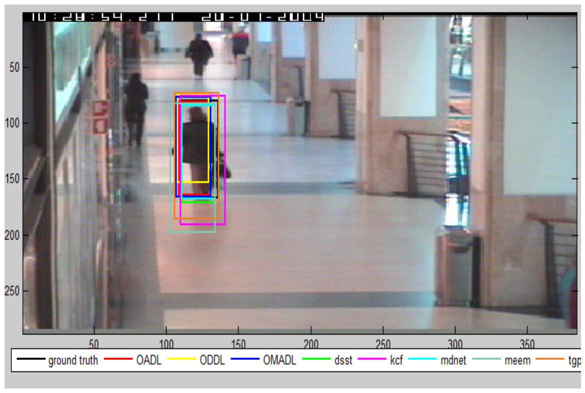
4.2 Qualitative Comparison with other trackers

To visually compare the results of our proposed trackers with other trackers, we show the results of trackers in the form of bounding boxes on frames of some video sequences from OTB-50 dataset. We choose some sequences like **Walking2**, **Football**, **Jumping**, **Singer1**, **Singer2** and we choose some of the challenging frames in which the effect of various conditions like Occlusion, background clutter, fast motion etc is clearly depicted. Our tracker performs consistently in all these video sequences under all the challenging conditions.

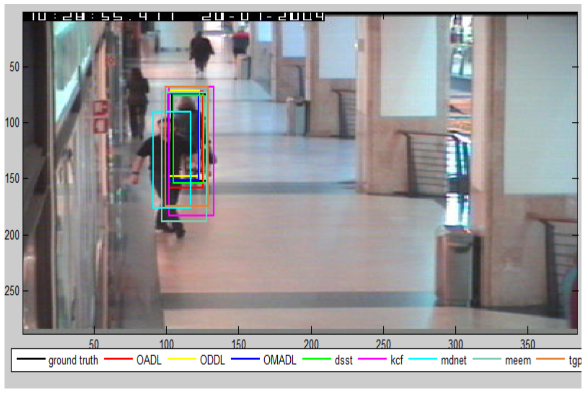
In the sequence **Singer2**, we also show a comparison between our two proposed algorithms. Clearly our proposed algorithm with multiple analysis dictionaries **OMADL**, has an improved performance than our proposed algorithm with a single analysis dictionary **OADL**.



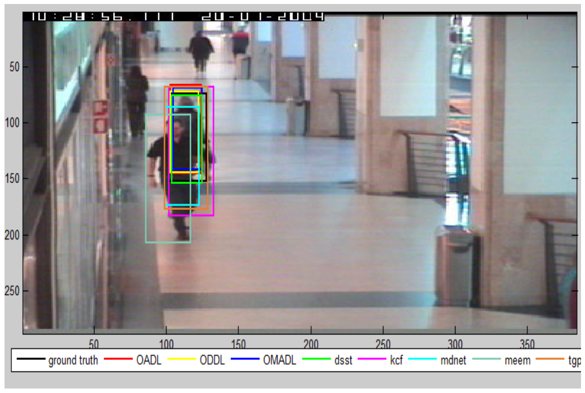
(a) Frame No.-69



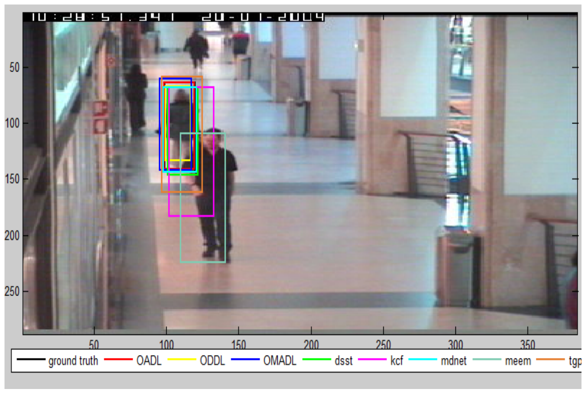
(b) Frame No.-152



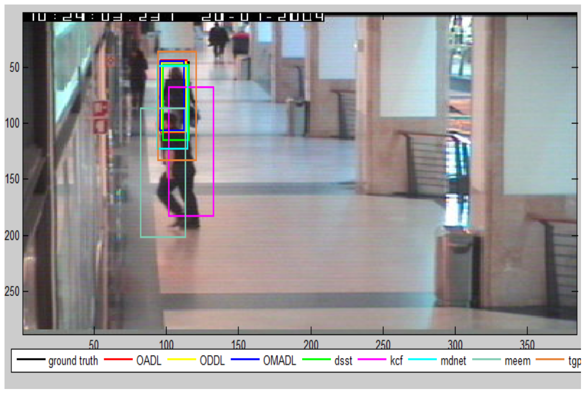
(c) Frame No.- 193



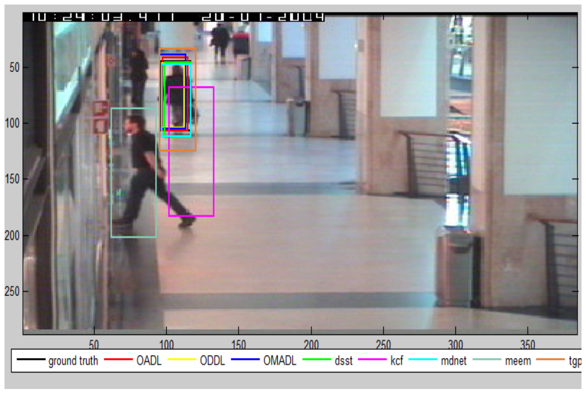
(d) Frame No.- 198



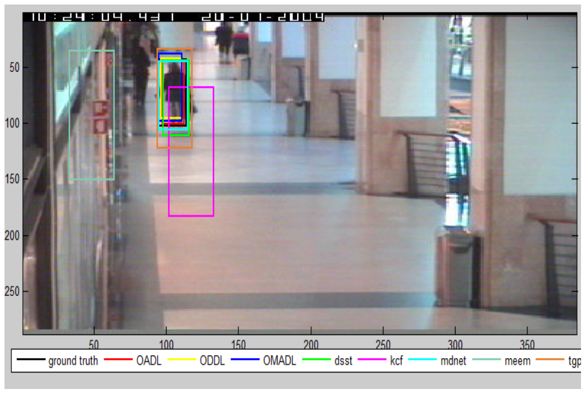
(e) Frame No.- 230



(f) Frame No.- 377

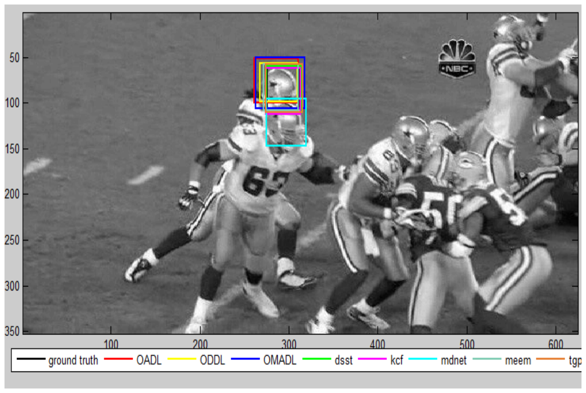


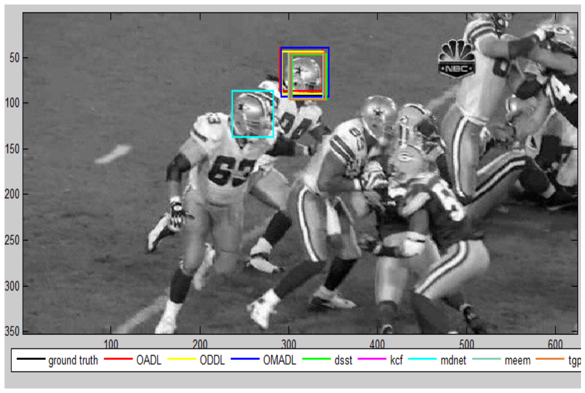
(g) Frame No.- 382

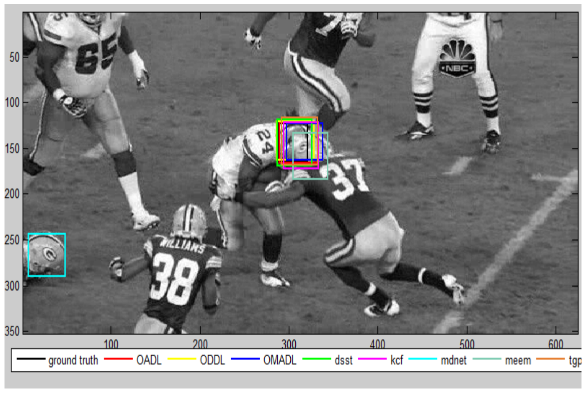


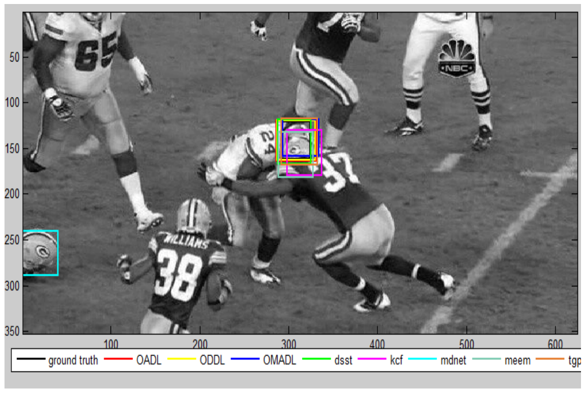
(h) Frame No.- 407

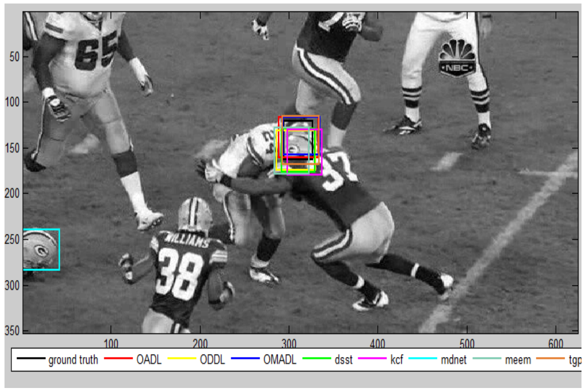
Figure 4.2.1: Video Sequence:Walking2

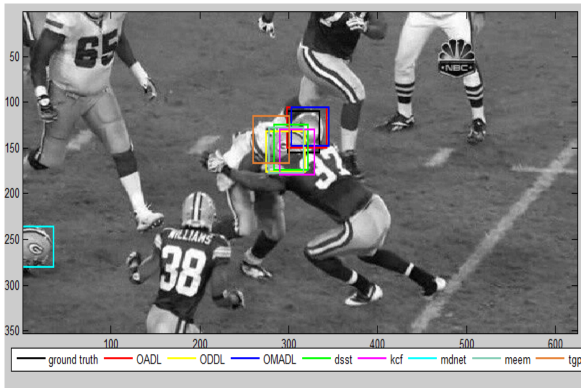

(a) Frame No.-61

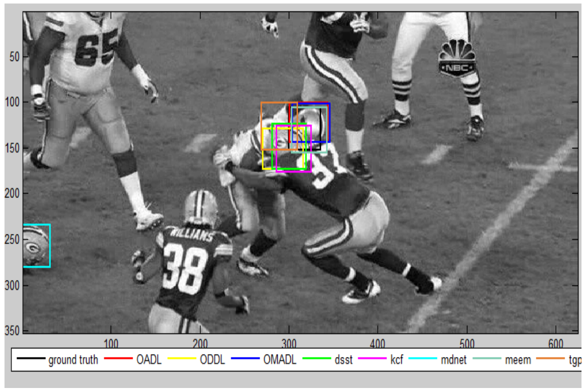

(b) Frame No.-70


(c) Frame No.- 283


(d) Frame No.- 286


(e) Frame No.- 287


(f) Frame No.- 290


(g) Frame No.- 291

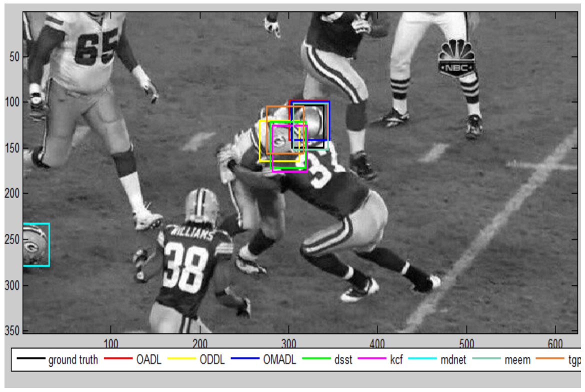
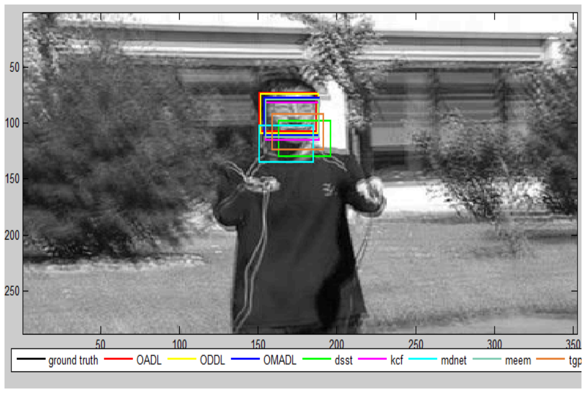
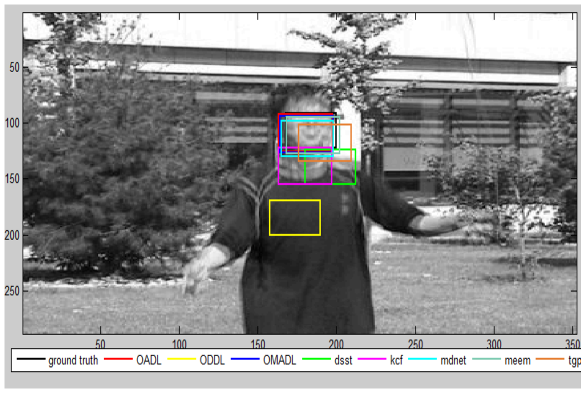

(h) Frame No.- 292

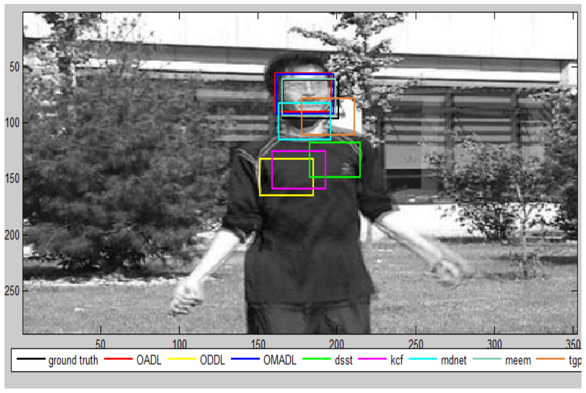
Figure 4.2.2: Video Sequence:Football



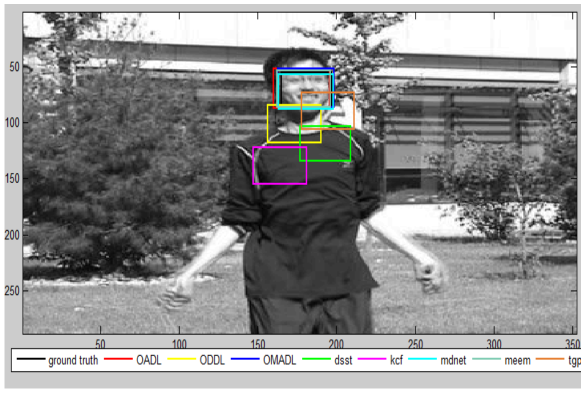
(a) Frame No.-16



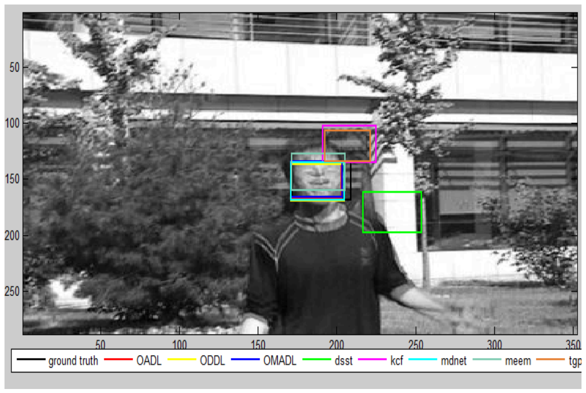
(b) Frame No.-35



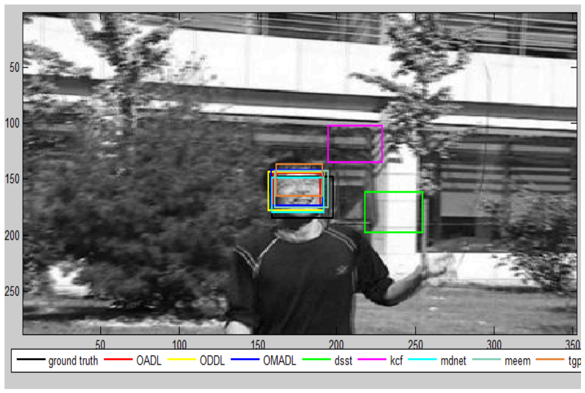
(c) Frame No.- 36



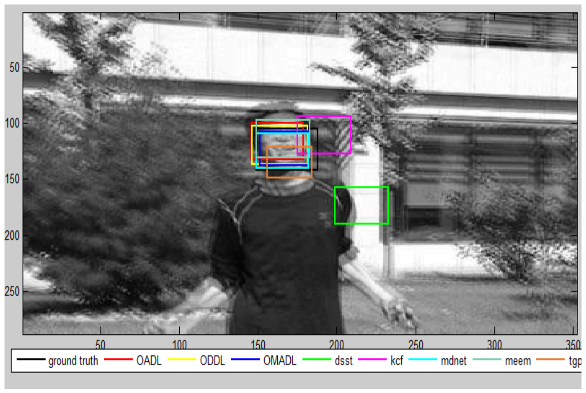
(d) Frame No.- 37



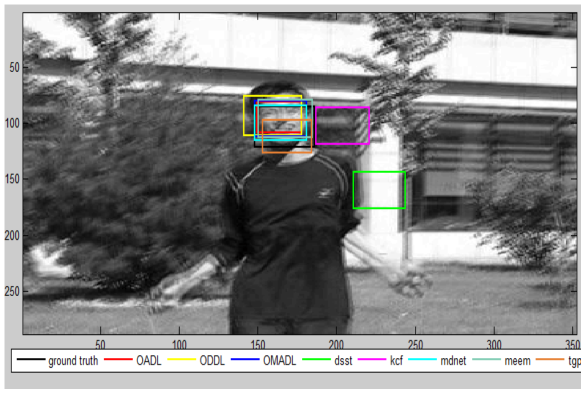
(e) Frame No.- 98



(f) Frame No.- 126

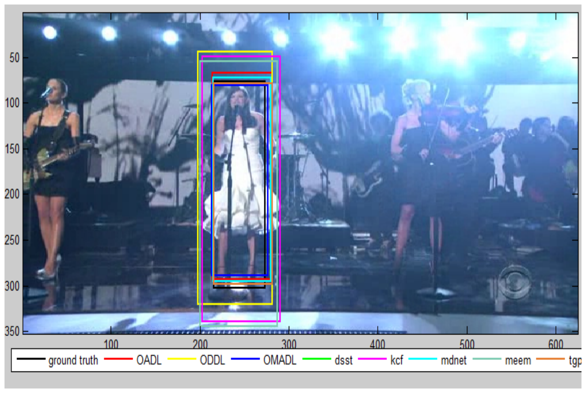


(g) Frame No.- 178

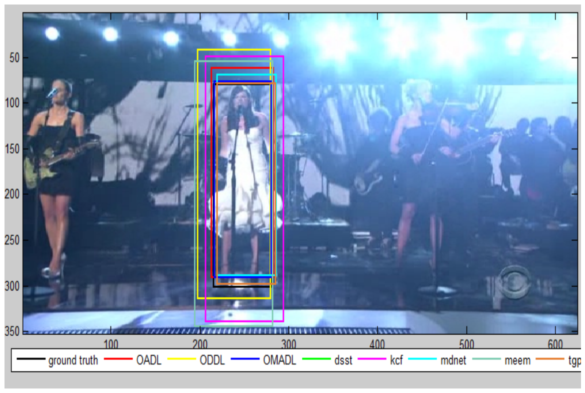


(h) Frame No.- 287

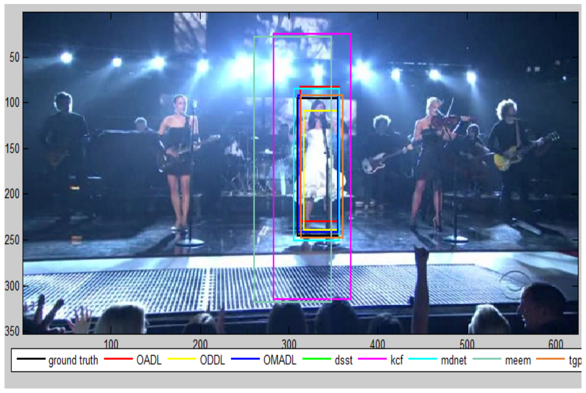
Figure 4.2.3: Video Sequence:Jumping



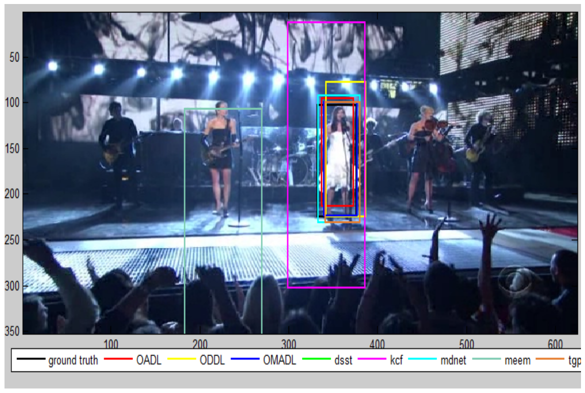
(a) Frame No.-67



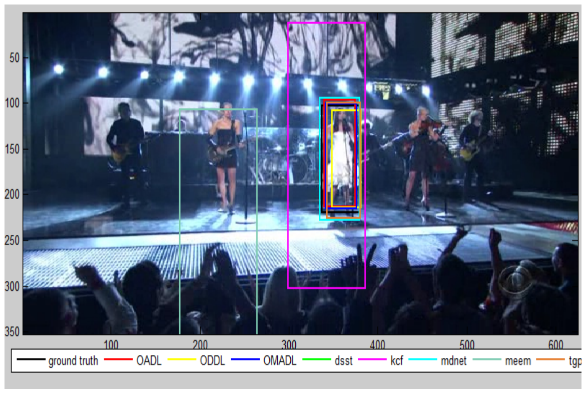
(b) Frame No.-70



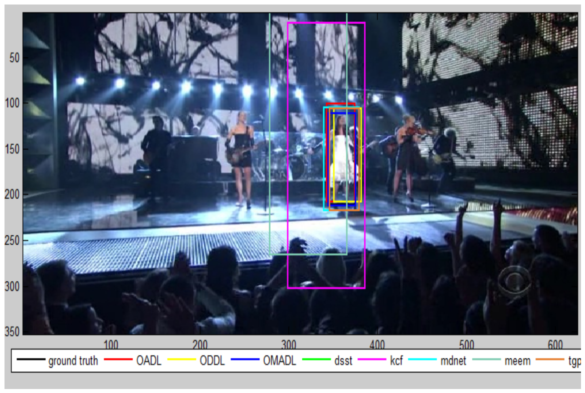
(c) Frame No.- 169



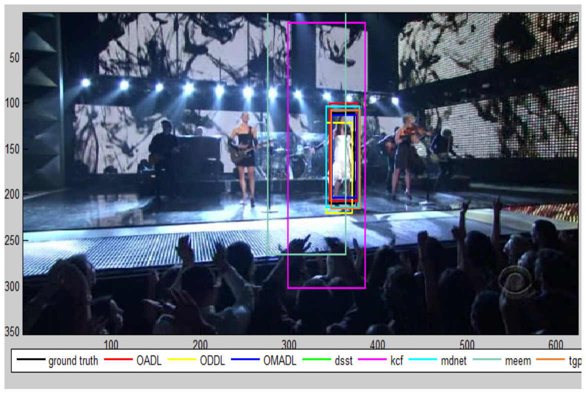
(d) Frame No.- 218



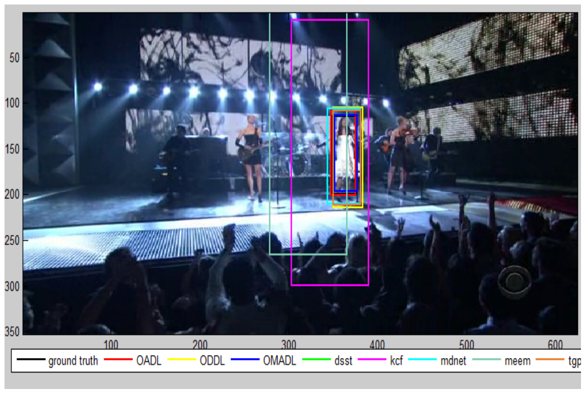
(e) Frame No.- 228



(f) Frame No.- 267

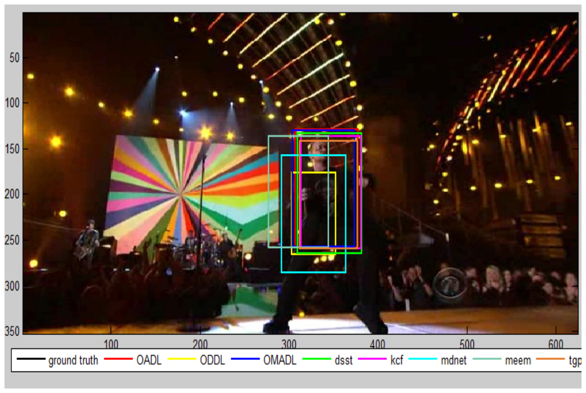


(g) Frame No.- 275

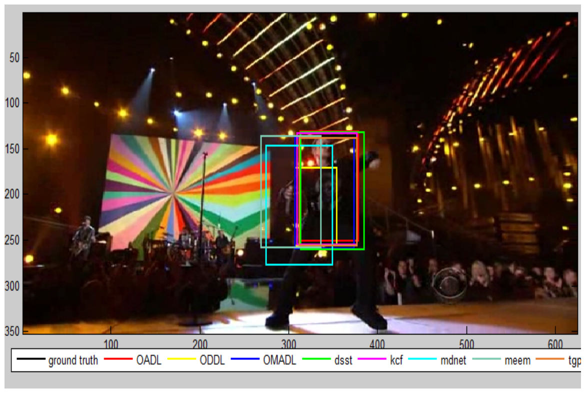


(h) Frame No.- 295

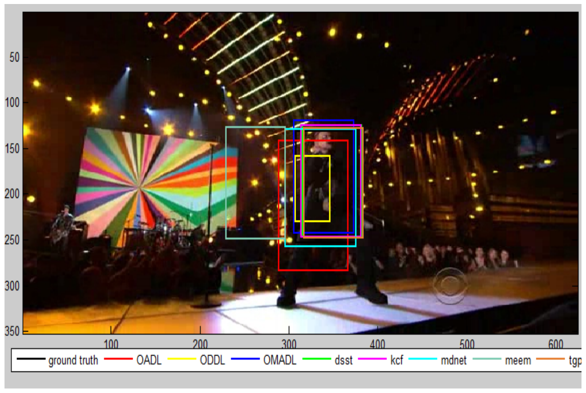
Figure 4.2.4: Video Sequence:Singer1



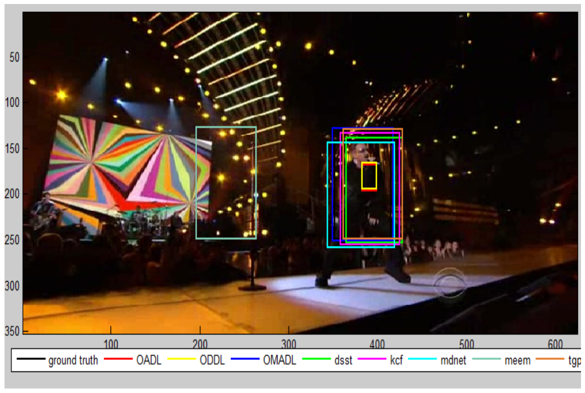
(a) Frame No.-16



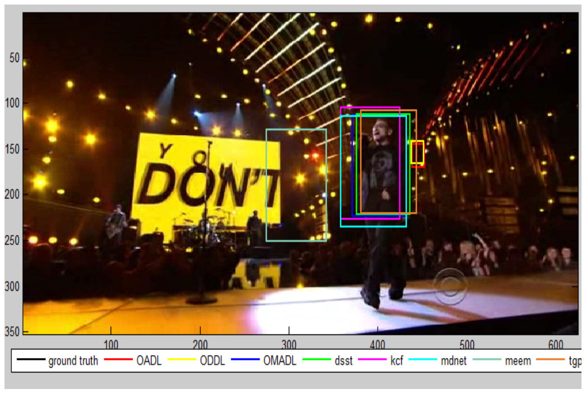
(b) Frame No.-20



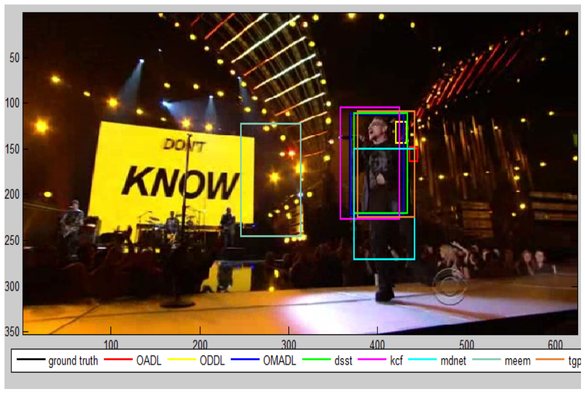
(c) Frame No.- 37



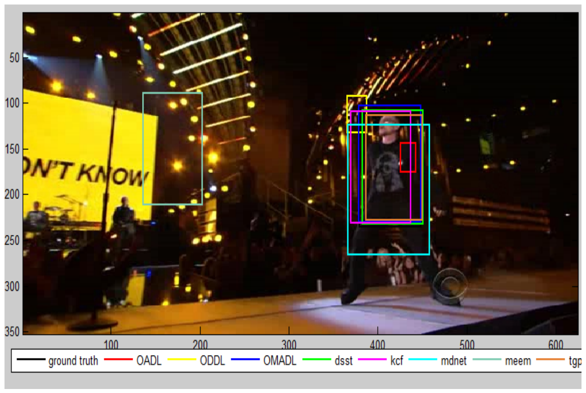
(d) Frame No.- 178



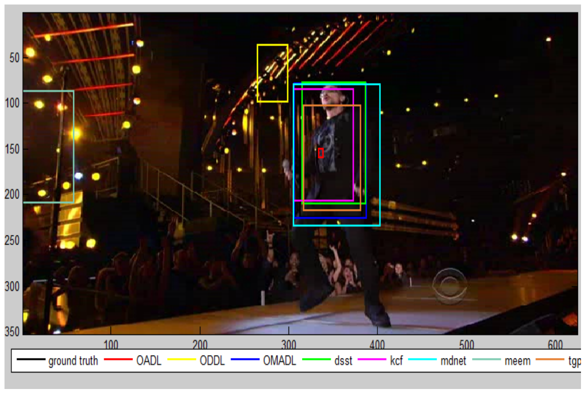
(e) Frame No.- 249



(f) Frame No.- 260



(g) Frame No.- 286



(h) Frame No.- 323

Figure 4.2.5: Video Sequence:Singer2

We compare our trackers with other state of the art trackers based on the frames illustrated above. The target in video Walking2 is influenced by occlusion and the target itself undergoes scale variation and low resolution. In frames 69 and 152, there is a scale variation in the target, both our trackers OADL and OMADL are able to track the target correctly and are closest to the ground truth annotation. In frames 193 and 198, when the target is occluded by a man, MEEM, TGPR, KCF and MDNET trackers lose the target partially but our trackers are able to track the target correctly. In frames 230, 377 and 382, MEEM and KCF have completely lost the target and they start tracking the occlusion. Our trackers are still tracking the target. Finally in frame 407, MEEM and KCF have lost the target and TGPR has not been able to adapt to the scale variation in the target. Both of our proposed trackers are still closest to the ground truth bounding box.

In the video Football, in frame 61, there is a partial occlusion in the target and MDNET starts tracking the occlusion. In all the future frames it loses the target completely and drifts with the occlusion. Our trackers tracks the target well. In frame 286, there is a occlusion again, the trackers MEEM and KCF start tracking the occlusion. In frame 287, all the trackers except both of our trackers and TGPR, are tracking the occlusion. In frame 290, all the trackers except both of our trackers, drift away from the target completely. Finally in frames 291 and 292, our trackers are still able to track the target. The bounding box of tracker MEEM comes back on the target in both these frames. But rest of the trackers have completely lost the target. Throughout the entire sequence, both of our trackers are able to track the target well.

In the video sequence Jumping, fast motion and motion blur occur in the target region. In frame 16, MDNET, TGPR and DSST lose the target partially. In frame 35, ODDL loses the target completely. In the same frame TGPR and KCF lose the target partially. In frame 36, ODDL, KCF and DSST have lost the target completely whereas MDNET and TGPR lose the target partially. In frame 37, MDNET is able to track the target back but ODDDL, TGPR, DSST and KCF have lost the target. In frame 98, DSST, TGPR and KCF are not able to track the target. By frame 126, TGPR has been able to come back to the target, but DSST and KCF have lost the target completely. In frame 178 and 287, TGPR is not able to track the target properly. Throughout the entire sequence both of our trackers stay close to the ground truth bounding box annotation and are able to track the target correctly.

In the next video sequence Singer1, illumination variation, occlusion and background clutter affect the target externally. The target itself goes through scale variation, deformation and out of plane rotation. In frames 67 and 70, the trackers MEEM, ODDL and TGPR are not able to model scale variation in the target properly. In frame 169, MEEM and TGPR have expanded much more than the ground truth bounding box and ODDL has shrunk a lot in comparison to the ground truth bounding box. In frame 218 and 228, MEEM loses the target completely and the size of KCF bounding box is very large as compared to the ground truth. In frames 275 and 295, MEEM is able to partially track the target but the size of the bounding boxes of MEEM and KCF is still very large as compared to ground truth. In the entire sequence, both of our trackers are able to track the target accurately.

In video sequence Singer2, the target is affected externally by illumination variation and background clutter. The internal variations that occur in the target in this sequence are deformation, in plane rotation and out of plane rotation. In this sequence in addition to comparison with other trackers, we compare the performance of our trackers OADL and OMADL against each other as well. In frames 16 and 20, the trackers MEEM, MDNET and ODDL are not able to track the target properly. On the contrary, both of our trackers are able to track the target accurately in frames 16 and 20. In all the subsequent frames till the last frame, MEEM loses the target completely. In frame 37, our tracker OADL, is only able to track the target partially. Furthermore, the bounding box of ODDL tracker is very small as compared to the ground truth bounding box. Whereas our tracker OMADL, is still able to track the target well. In frame 178, the bounding boxes of our proposed tracker OADL and the bounding box of the tracker ODDL shrink a lot in comparison to the ground truth. Our tracker OMADL still tracks the target correctly. In frame 249, OADL and ODDL have lost the target completely. In frame 260 and 286, the tracker MDNET, tracks the target partially. Our tracker OMADL tracks the target well. In frame 323, TGPR tracker's bounding box becomes quite small as compared to ground truth and is not able to track the target accurately. Our tracker OMADL is able to track the target well throughout the entire video sequence. Whereas our tracker OADL is not able to track the target properly in this sequence. Thus, our approach with multiple analysis dictionaries (OMADL) shows an improvement over our work with single analyis dictionary (OADL).

4.3 Comparison on the basis of Speed

In terms of frames per second our algorithm using single analysis dictionary gives a tracking speed of 5 frames per second and our algorithm using multiple analysis dictionaries gives a speed of 3 frames per second. Whereas, trackers that use deep learning architectures like MDNET has a tracking speed of 1 frame per second and Convolutional Neural Net Bagging for Online Visual Tracking(CNN Bagging) [45] has a tracking speed of around 1.6 frames per second. This makes our algorithm OMADL three times faster than MDNET and around two times faster than CNN Bagging. Moreover, the model update in OMADL is simpler and faster as compared to MDNET and CNN Bagging. This makes our algorithm more desirable for online tracking applications.

4.4 Conclusion

Thus, in this thesis we propose two tracking algorithms. The first one uses single analysis dictionary to model the target and the second one uses multiple analysis dictionaries to model the target. Our work is so far to the best of our knowledge the first work that uses analysis dictionaries for visual tracking and analysis dictionary approaches clearly show an improvement over synthesis dictionary approaches for visual tracking. In our approach with multiple analysis dictionaries, we dynamically weigh different features. This way the feature which has a higher capability to describe the target gets higher weight as compared to other features in the representation. In an experimental evaluation on OTB-50 dataset that contains videos that are influenced by the combinations of all the tracking situations, we are able to demonstrate that our proposed method using multiple analysis dictionaries (OMADL) is able to outperform many state of the art trackers.

4.5 Future Work

For future work, to deal with a condition on which our model gives average performance, we can add more dictionaries to the appearance model of our multiple analysis dictionaries algorithm. These dictionaries can be learnt on such features that are invariant to that particular condition out of the 11 conditions on which our algorithm has an average performance. Then we can combine this dictionary with other dictionaries using the same weighting measure explained in Section (2.2.4). For example, to deal with rotations, we can further add a dictionary learnt on a feature that is rotation invariant to the appearance model of our proposed algorithm that uses multiple analysis dictionaries. We will then combine this dictionary with other dictionaries using the same weighting measure as explained in Section (2.2.4).

Bibliography

[1] F. Yang, Z. Jiang, and L. S. Davis, "Online discriminative dictionary learning for visual tracking," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on.* IEEE, 2014, pp. 854–861.

[2] X. Jia, H. Lu, and M.-H. Yang, "Visual tracking via adaptive structural local sparse appearance model," in *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 1822–1829.

[3] B. Liu, J. Huang, C. Kulikowski, and L. Yang, "Robust visual tracking using local sparse appearance model and k-selection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2968–2981, 2013.

[4] X. Mei and H. Ling, "Robust visual tracking using 1 minimization," in *Computer Vision, 2009 IEEE 12th International Conference on.* IEEE, 2009, pp. 1436–1443.

[5] S. Zhang, H. Yao, H. Zhou, X. Sun, and S. Liu, "Robust visual tracking based on online learning sparse representation," *Neurocomputing*, vol. 100, pp. 31–40, 2013.

[6] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Low-rank sparse learning for robust visual tracking," in *European Conference on Computer Vision*, 2012, pp. 470–484.

[7] ——, "Robust visual tracking via multi-task sparse learning," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 2042–2049.

[8] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparsity-based collaborative model," in *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1838–1845.

[9] R. Liu, X. Lan, P. C. Yuen, and G. Feng, "Robust visual tracking using dynamic feature weighting based on multiple dictionary learning," in *24th European Signal Processing Conference, EUSIPCO 2016, Budapest, Hungary*, 2016, pp. 2166–2170.

[10] M. Yaghoobi, S. Nam, R. Gribonval, and M. E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Transactions on Signal Processing*, vol. 61, no. 9, pp. 2341–2355, 2013.

[11] H. Fan and J. Xiang, "Robust visual tracking with multitask joint dictionary learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 5, pp. 1018–1030, 2017.

[12] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[13] J. Zhang, S. Ma, and S. Sclaroff, "MEEM: robust tracking via multiple experts using entropy minimization," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2014.

[14] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[15] J. Gao, H. Ling, W. Hu, and J. Xing, "Transfer learning based visual tracking with gaussian processes regression," in *European Conference on Computer Vision*. Springer, Cham, 2014, pp. 188–203.

[16] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[17] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, “Struck: Structured output tracking with kernels,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[18] Z. Kalal, J. Matas, and K. Mikolajczyk, “Pn learning: Bootstrapping binary classifiers by structural constraints,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 49–56.

[19] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[20] M. Aharon, M. Elad, and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[21] A. Vedaldi and B. Fulkerson, “VLFeat: An open and portable library of computer vision algorithms,” http://www.vlfeat.org/, 2008.

[22] B. Zhang, Y. Gao, S. Zhao, and J. Liu, “Local derivative pattern versus local binary pattern: face recognition with high-order local pattern descriptor,” *IEEE transactions on image processing*, vol. 19, no. 2, pp. 533–544, 2010.

[23] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[24] B. Babenko, M.-H. Yang, and S. Belongie, “Visual tracking with online multiple instance

learning," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 983–990.

[25] D. Wang, H. Lu, and M.-H. Yang, "Least soft-threshold squares tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2371–2378.

[26] S. He, Q. Yang, R. W. Lau, J. Wang, and M.-H. Yang, "Visual tracking via locality sensitive histograms," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2427–2434.

[27] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *European conference on computer vision.* Springer, Berlin, Heidelberg, 2012, pp. 864–877.

[28] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Computer Vision–ECCV 2012.* Springer Berlin/Heidelberg, 2012.

[29] Y. Wu, B. Shen, and H. Ling, "Online robust image alignment via iterative convex optimization," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 1808–1814.

[30] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan, "Locally orderless tracking," *International Journal of Computer Vision*, vol. 111, no. 2, pp. 213–228, 2015.

[31] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust l1 tracker using accelerated proximal gradient approach," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 1830–1837.

[32] L. Sevilla-Lara and E. Learned-Miller, "Distribution fields for tracking," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 1910–1917.

[33] J. Kwon and K. M. Lee, "Tracking by sampling trackers," in *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1195–1202.

[34] T. B. Dinh, N. Vo, and G. Medioni, "Context tracker: Exploring supporters and distracters in unconstrained environments," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE, 2011, pp. 1177–1184.

[35] J. Kwon and K. M. Lee, "Visual tracking decomposition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 1269–1276.

[36] S. Stalder, H. Grabner, and L. Van Gool, "Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition," in *Computer vision workshops (ICCV Workshops), 2009 IEEE 12th international conference on.* IEEE, 2009, pp. 1409–1416.

[37] H. Grabner, C. Leistner, and H. Bischof, "Semi-supervised on-line boosting for robust tracking," *Computer Vision–ECCV 2008*, pp. 234–247, 2008.

[38] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International Journal of Computer Vision*, vol. 77, no. 1, pp. 125–141, 2008.

[39] H. Grabner and H. Bischof, "On-line boosting and vision," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 260–267.

[40] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 798–805.

[41] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

[42] R. T. Collins, "Mean-shift blob tracking through scale space," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003, pp. II–234.

[43] D. Comaniciu and P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, 2003.

[44] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Eur. Conf. on Computer Vision (ECCV 2002).* Springer Berlin Heidelberg, 2002.

[45] H. Li, Y. Li, and F. Porikli, "Convolutional neural net bagging for online visual tracking," *Computer Vision and Image Understanding*, vol. 153, pp. 120–129, 2016.