Object Tracking Using a Multi-Feature Joint Sparse Representation

ELENE6876.2020Spring. Course Project report
Ali Zare az2584, *Columbia University*

Abstract

Existing object tracking algorithms based on sparse representation have depended only on a single feature of the target object. However, various features extracted from the same objects tend to share some commonalities while having their own specific, unique representation. The shared commonalities can be used to achieve better and more accurate appearance models. To address this issue, a sparse reconstruction algorithm was introduced in [1]. In the paper, they propose a tracking algorithm based on a multi-feature joint sparse representation. The templates for the sparse representation can include pixel values, textures, and edges. In the multi-feature joint optimization, noise and occlusion are dealt with using a separate set of templates for noise and adaptive template dictionary that is changed based on the relevance of templates to the sample candidate. In this paper, they showed that by relying on multi-feature representation, the appearance model can perform better in object tracking. In an attempt to replicate this paper, we implemented a similar object tracking algorithm with some modifications and tested the performance of the algorithm on two publicly available datasets. [2].

1. Introduction

Visual object tracking is usually after predicting moving objects' states, e.g. location, scale, or velocity, from the observations in a video. The importance of performing this task with high precision is crucial in many applications, some examples of which are visual surveillance, vision-based control, human-computer interfaces, intelligent transportation, and augmented reality. Object tracking algorithms can be broken down into two components: A motion model, and an appearance model. illumination, viewpoint and pose, occlusions, and background clutter are some of the known factors that can make it harder to come up with a robust appearance model that is capable of adapting to changes in the background. Many appearance models [3],[4], [5],[6] have been proposed for object tracking. These include models based on histograms, kernel density estimates, Gaussian mixture models (GMMs), conditional random fields, subspaces, and discriminative classification.

Sparse representation-based object appearance models [5, 6] have long been used for object tracking purposes[5,6,7]. Conventionally sparse approximation trackers include an L1-regularizer in their objective function, guaranteeing sparsity in the approximation. An observation in a frame can be sparsely represented in the space spanned by object templates and trivial templates. The object templates are usually obtained from the previous frames since videos are temporally highly correlated. The trivial templates account for the pixels contaminated by occlusion or noise. The conventional sparse representation-based tracking only uses the pixel gray levels in the sequence of frames to construct the template set. The pixel gray level is not representative enough to distinguish the object from the background or from other objects. It is necessary to construct a new and more powerful sparse representation-based appearance model that can combine useful features, such as colors, textures, or edges.

In [1], they proposed a tracking algorithm based on a multi-feature joint sparse representation which is used to combine the templates constructed from the different image features. The candidate template is then reconstructed by a linear combination of the template dictionary. The observation with the smallest reconstruction error is then chosen as the tracking result under the motion model.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

In the original paper [1], a new particle's performance in object tracking was assessed by reconstruction of the new template from a linear combination of the previous templates. A tracking result is an image patch that is normalized to a particular size, flattened, and stacked to produce a "u" dimensional vector, where "u" equals to the number of the pixels in the normalized patch. Let H=[h1,h2,...,hn] be a set of n templates where m can be written as a linear combination of, plus a residual term which in itself consists of u templates each representing a pixel in templates. We can write:

$$m = \sum_{i=1}^{n=number\ of\ templates} w_i \boldsymbol{h_i} + \varepsilon = \boldsymbol{HW} + \varepsilon$$

$$\varepsilon = [\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_u}][e_1, e_2, \ldots, e_u]^T = \boldsymbol{Te}$$

$$\boldsymbol{m} = [\boldsymbol{H}\ \boldsymbol{T}][\boldsymbol{w}^T\ \boldsymbol{e}^T]^T = \boldsymbol{BF} \qquad \text{eq1}$$

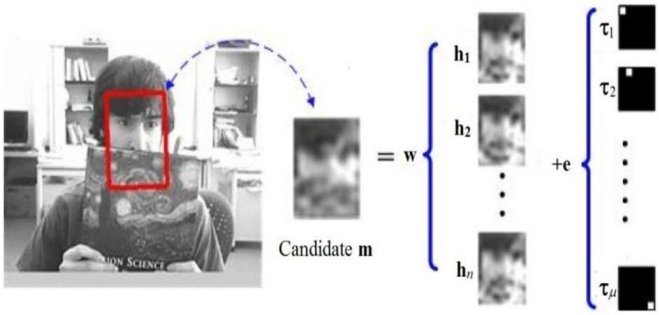

Figure 1. An illustration of the appearance model. The figure is taken from [1].

Expanding this to multiple features we arrive at:

$$m^k = \sum_{i=1}^{n=number\ of\ templates} w_i^k \boldsymbol{h_i^k} + \varepsilon^k = \boldsymbol{H^k W^k} + \varepsilon^k$$

We are after the F that minimizes the below objective function:

$$\hat{\mathbf{F}} = \min_{\mathbf{F}} \left(\frac{1}{2} \sum_{k=1}^{K} \left\| \mathbf{m}^k - \mathbf{B}^k \mathbf{f}^k \right\|_2^2 + \lambda \left(\sum_{i=1}^{n} \left\| \mathbf{D}_i \mathbf{w}_i \right\|_2 + \sum_{j=1}^{\mu} \left\| \mathbf{e}_j \right\|_2 \right) \right),$$

Where L2 is done over the various features of a template and L1 is the norm over various templates. Norm 2 enforces that if a template has poor contribution in the reconstruction of a new template then all of its weights in the feature domain must be close to zero.
D is the metric that determines how much a template is close to the test template:

Table 1: The multi-feature joint sparse representation for an observation

1: Input: templates $\{\mathbf{H}^k\}_{k=1}^K$, an observation $\{\mathbf{m}^k\}_{k=1}^K$, the regularization parameter λ, and the step size value α.
2: Initialization: Initialize $\mathbf{f}^{k,0}$ and $\mathbf{v}^{k,0}$; Set $\gamma_0 = 1$ and $t \leftarrow 0$.
3: Repeat {Main loop}
4: $\mathbf{f}^{k,t+1} \leftarrow \tilde{\mathbf{d}}^k \odot (\mathbf{v}^{k,t} - \alpha(-(\mathbf{B}^k)^T \mathbf{m}^k + (\mathbf{B}^k)^T (\mathbf{B}^k)\mathbf{v}^{k,t}))$, $k = 1, 2, ..., K$
5: $\mathbf{f}_i^{t+1} \leftarrow \max\left\{1 - \frac{\alpha\lambda}{\left\|\mathbf{f}_i^{t+1}\right\|_2}, 0\right\} \mathbf{f}_i^{t+1}$, $i = 1,2, \ldots, n$
6: $\gamma_t = \frac{2}{t+2}$
7: $\mathbf{V}^{t+1} \leftarrow \mathbf{F}^{t+1} + \frac{\gamma_{t+1}(1-\gamma_t)}{\gamma_t}(\mathbf{F}^{t+1} - \mathbf{F}^t)$
8: $t \leftarrow t+1$,
9: until convergence is reached
10: Output: $\mathbf{F}$.

$$\mathbf{D}_i = diag\left(\left\| \tilde{\mathbf{m}}^1 - \mathbf{h}_i^1 \right\|, \left\| \tilde{\mathbf{m}}^2 - \mathbf{h}_i^2 \right\|, ..., \left\| \tilde{\mathbf{m}}^k - \mathbf{h}_i^k \right\|, ..., \left\| \tilde{\mathbf{m}}^K - \mathbf{h}_i^K \right\| \right)$$

To minimize the objective function, they used accelerated gradient descent and followed the algorithm in Table 1 (taken from [1]).

For the motion model, the particles were generated from a gaussian distribution. The random variable sampled from the gaussian distribution was a vector z=[x, y, r,c, a, phi] where x and y correspond to the location of the center of the rectangular template and (r,c, a, phi) are the deformation parameters, corresponding to the rotation angle, the scale, the aspect ratio, and the skew direction, respectively. The Gaussian distribution was centered at the previous frame's z and its sigma was chosen arbitrarily. The task of tracking is estimating the state of an object given the observation **m**. This can be represented as a Bayesian posterior probability inference:

$$p(\mathbf{z}_t \mid \mathbf{M}_t) \propto p(\mathbf{M}_t \mid \mathbf{z}_t) \int p(\mathbf{z}_t \mid \mathbf{z}_{t-1}) p(\mathbf{z}_{t-1} \mid \mathbf{M}_{t-1}) d\mathbf{z}_{t-1}$$

and

$$p(\mathbf{M}_t \mid \mathbf{z}_t) \propto \exp\left\{ -\eta \sum_{k=1}^{K} \theta^k \left\| \mathbf{m}_t^k - \mathbf{H}^k \mathbf{w}^k \right\|_2^2 \right\}$$

Table 2 (taken from [1]) shows the algorithm for particle selection and motion model.

Table 2: The tracking Algorithm

1: Input the set of the templates $\{\mathbf{H}^{k}\}_{k=1}^{K}$ which has been updated at the previous frame *t*-1, and the state $\mathbf{z}_{t-1}$ of the object at frame *t*-1.
2: Use the dynamic model $G(\mathbf{z}_{t-1},\Sigma)$ to produce a number of particles at the current frame *t*.
3: Use the algorithm shown in Table 1 to estimate the multi-feature joint sparse representation for each particle $\mathbf{z}_t$ at the current frame *t*.
4: Evaluate the weight of each particle $\mathbf{z}_t$ using $p(\mathbf{M}_t \mid \mathbf{z}_t)$ in (22).
5: Take the observation specified by the particle with the largest weight as the tracking result.
6: The set of the templates is updated using the method in Section 4.4.
7. Output the tracking result at the current frame and the new set of templates.

3. Methodology

We sought out to implement the object tracking algorithm in [1] in its entirety. The overall framework consists of an appearance model (Tabel 1) and a motion model (Tabel 2). To simplify the model, the particles were limited to 2 dimensions z=[x,y], where x and y are the location of the rectangular for object tracking. The size and angle of the rectangular were kept constant.

3.1. Objectives and Technical Challenges

One of the main challenges was the long-running time that the algorithm needed to run for kitesurfer and bolt datasets. This was due to a large number of pixels in templates. To shorten the running time of the algorithm the input images were downsampled and rescaled by a factor of 0.3.

Another factor was choosing a regularization parameter that would minimize the appearance model error while keeping the algorithm time efficient. We found lambda equal to 1e-5 to be a good balance point between minimum error and efficient running time.

3.2. Problem Formulation and Design

Similarly to the paper, we introduced a threshold for updating the templated dictionary. Whenever the difference between the tracking result **m** and a template was greater than a threshold the tracking result would replace the template with the maximum error in the template dictionary. This was done to keep the templates relevant to the changes that happen during the video.

The threshold was set to 60% of the tracking results of the Frobenius norm. Unlike the paper, we decided to use simple gradient descent with soft thresholding instead of accelerated gradient descent due to convergence issues.

We also did not include the d factor in table 1, step 4.

4. Implementation

The algorithm was implemented in Matlab. To enhance the performance of object tracking algorithms, we used multiple features.

For each frame, the green, red, and blue channels alongside the edge picture of the grayscale picture of the frame were introduced as the feature vectors.

Edge pictures were found by looking for zero-crossings after filtering the grayscale image with a Laplacian of Gaussian (LoG) filter. Figure 2 shows the picture of a sample frame for each feature.

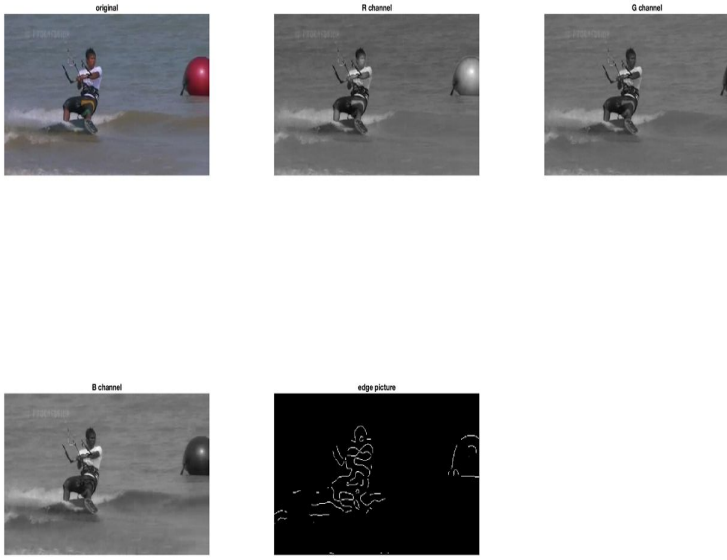

Figure 2. Top Left: original frame. Top middle: the red channel of the picture. Top Right: the green channel. Bottom left: the blue channel. Bottom Right: the edge picture.

We used the publicly available datasets[2] to evaluate the performance of the algorithm.

4.1. Performance of the appearance model

Figure 3 shows the performance (loss value vs iterations) of the appearance model for multiple z particles and the convergence of the gradient descent

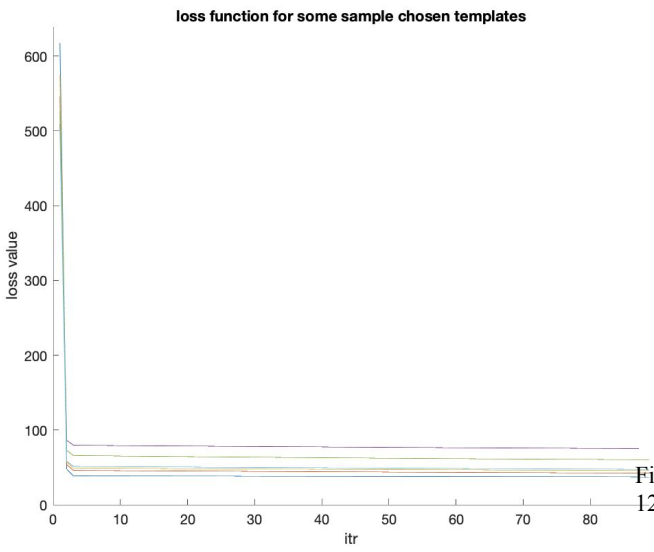

Figure 3. Loss value vs Iterations.

5. Results

5.1. Project Results

The results for object tracking can be seen in figure 4-8. To obtain these results the initial 59 frames of the video were kept as the reference frames for template dictionaries. The number of z particles generated in the motion model was set to 200 particles.

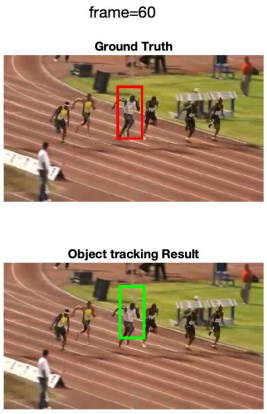

Figure 4. Ground truth and result of object tracking for frame 60 in Bolt, Zhang dataset.

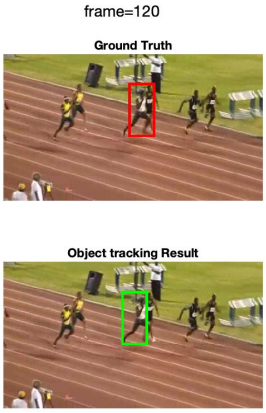

Figure 5. Ground truth and result of object tracking for frame 120 in Bolt, Zhang dataset.

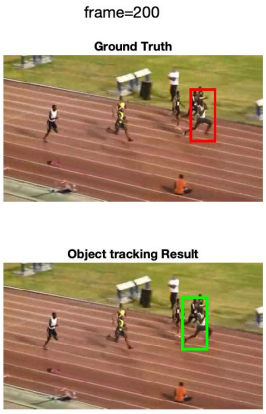

Figure 6. Ground truth and result of object tracking for frame 200 in Bolt, Zhang dataset.

As can be seen from figures 4-6 the introduction of RGB channels as features to the model prevents the tracker from following the wrong runner.

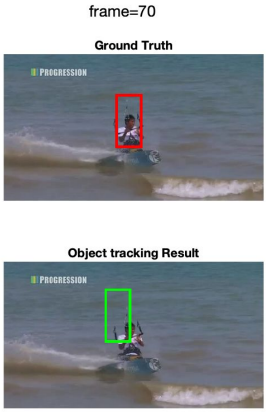

Figure 7. Ground truth and result of object tracking for frame 70 in Kitesurfer, Zhang dataset.

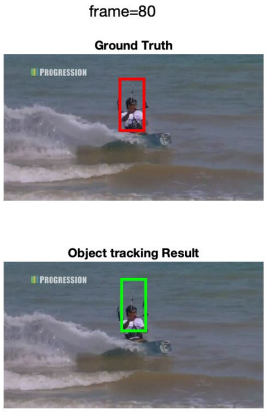

Figure 8. Ground truth and result of object tracking for frame 80 in Kitesurfer, Zhang dataset.

We can see good results for the kitesurfer dataset as well. In this dataset, the goal was tracking the head of the kitesurfer.

5.3. Discussion of Insights Gained

5.3.1 The effect of using multiple features vs single feature:

Using multiple features for object tracking is helpful but computationally expensive (depending on how many features we include for tracking purposes for each frame). As a result, it is important to know when and what features are best to be used depending on the dataset. For example in datasets where there are multiple objects similar to the target object using multiple features such as color, texture, and edge features can be helpful in distinguishing the target object from other objects. However, in applications where there is one distinct object from the background, this might not be computationally beneficial. One way to go about solving the computational expense would be to resize the images and downsample them.

5.3.2. Effect L1 regularizer parameter

Our results indicate that finding the right lambda parameter is crucial in object tracking. Depending on the lambda different sparse events can be included in the appearance model. The smaller the lambda the less sensitive will the appearance model be to the changes between the templates. The larger the lambda the more important the difference between the templates will be. Resulting in singling out one template versus other templates in the template dictionary for the linear combination of templates. Lambda parameter has to be chosen in a way that a few of the templates in the template dictionary have dominating coefficients while others have small coefficients. However, choosing large lambdas might limit the performance of the tracker. We found that lambda 1e-5 seems to do a good job. The lambda value was found experimentally.

5.3.3. Effect of various gradient descent methods

We realized that our model was quite sensitive to the size of the steps in gradient descent. Therefore using accelerated gradient descent resulted in divergence instead of convergence. To improve the convergence, we found the step size alpha for each specific feature instead of choosing a single alpha for all features:

$F_{n+1}^{k} = Prox(F_{n}^{k} - alpha^{k} . grad(f))$

Where $alpha^{k} = 1/norm(B^{k}, fro)^{2}$

B^k is the stack of templates for feature k.

This helped the optimizer to adapt better for each feature.

5.3.4. Effect of Gaussian distribution variance in the generation of motion model particles.

The variance of the gaussian distribution in the motion model determines how far apart the new tracking samples should be from the previous tracking location. This is dependent on the frame rate and the speed of the object and how much its location changes locally. The smaller the gradient means the expected change in the location of the target object is small. This parameter has to be set based on the properties of the dataset.

6. Conclusion

We were able to successfully implement the object tracking algorithm based on sparse representation and multi-feature appearance models. The main advantages of this framework are 1) estimating appearance models that are less sensitive to noise because of the commonality of multi-features in the representation of the same object that they are extracted from and 2) the ability of the appearance model for adapting to changes in the environment and background of the object and distinguishing the object from other similar objects. We achieved relatively high tracking accuracies,

To further improve our results, one can include more distinctive features and use higher resolution images at the cost of computational power.

6. Acknowledgment

I would like to thank Dr. John Wright for giving extremely helpful advice on the project, as well as the course TA's for their responsiveness towards any questions.

7. References

GitHub code:
https://github.com/alizaree/ObjectTrackingSparse

[1] W. Hu, W. Li, X. Zhang, and S. Maybank, "Single and multiple object tracking using a multi-feature joint sparse representation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 4, pp. 816–833, Apr. 2015.

[2] Zhang publicly available dataset, access link: http://www4.comp.polyu.edu.hk/~cslzhang/CT/CT.htm

[3] X. Lan, S. Zhang, P. C. Yuen and R. Chellappa, "Learning Common and Feature- Specific Patterns: A Novel Multiple-Sparse-Representation-Based Tracker," in IEEE Transactions on Image Processing, vol. 27, no. 4, pp. 2022-2037, April 2018.

[4] Y. Wu, E. Blasch, G. Chen, L. Bai, and H. Ling, "Multiple source data fusion via sparse representation for robust visual tracking," in Proc. Int. Conf. Inf. Fusion, Jul. 2011, pp. 1–8.

[5] W. W. Zou, P. C. Yuen, and R. Chellappa, "Low-resolution face tracker robust to illumination variations," IEEE Trans. Image Process., vol. 22, no. 5, pp. 1726–1739, May 2013.

[6] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust Face Recognition via Sparse Representation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 31, no. 2, pp. 210-227, Feb. 2009.

[7] X. Mei and H. Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 33, no. 11, pp. 234-778, Nov. 2011.

[8] B.S. Manjunath, J.R. Ohm, V. Vasudevan, and A. Yamada, "Color and Texture Descriptors," IEEE Trans. on Circuits and Systems for Video Technology, vol. 11, no. 6, pp. 703-715, June 2001.

[9] F. Chen, Q. Wang, S. Wang, W. Zhang, and W. Xu, "Object Tracking via Appearance Modeling and Sparse Representation," Image and Vision Computing, vol. 29, no. 11, pp. 787-796, 2011.

[10] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust Online Appearance Models for Visual Tracking," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 25, no. 10, pp. 1296-1311, Oct. 2003.