CS 365 Project Report

Distribution Fields for Robust Object Tracking

Maninderjit Singh 10381

Akshay Agrawal 10058

Mentor- Amitabha Mukerjee

Abstract

In this project we explored a recent tracking algorithm based on distribution fields which gives results comparable to state of art. Previous algorithms used matching functions or templates to track object of interest in subsequent frame. To avoid oversensitivity of these templates to spatial features blurring techniques were used. As blurring leads to loss of intensity information, distributed fields tends to avoid this by splitting the image into n layer or bins each containing similar intensity pixels. This approach is combined with an online learning model of tracked object to produce good results in complex backgrounds with occlusion, illumination and appearance changes. We modified this algorithm to check for best model match at different scales to make this algorithm scale invariant.

Introduction

Majority of the Object Recognition Techniques in the current state of art converges to a simple algorithm of

1. Mapping the image to appropriate descriptors and then
2. Using the obtained descriptors to find correspondences between parts of the images.

Distribution Fields (DFs) is one such image descriptor that encompasses the important aspects of some other popularly known image descriptors like, SIFT, HOG, kernel-based descriptors, geometric blur etc. and thus presents the following properties of,

- robustness to image noise
- robustness to small misalignments
- smoothness to degradations (to spatial location, scale, brightness)
- smoothness in its representation and
- ability to adapt to different images easily.

The report hereafter is organized as:
Section 1 : Description of Distribution Fields
Section 2 : Framework of suggested algorithm
Section 3 : Comparison with original implementation
Section 4 : Conclusion and future work
Section 5 : References and Acknowledgement

Distribution Fields

What are DFs?

Distribution fields are an elaborate data structure that is obtained by quantizing the feature information of an image.

So, let's assume that an (NxM) size image is made up of pixels of values ranging over an interval 'B', then the distribution fields data structure obtained will be of dimension (NxMxk); where k is any number less than 'B'.

How are they constructed?

Distribution fields are constructed through this 2 step procedure of:
- Image Explosion and
- Image Smoothing.

Detailed description of the terms is mentioned below:

<u>Explosion:</u>

1. Explosion refers to the process of breakdown of the image in "layers".

2. Each layer has a value or a range of values associated with it.

3. All the pixels in the original image which had a value that falls in the value set of a certain layer are marked '1' in that layer and all the other pixels are marked zero. Mathematically,

$$d(i; j; k) = \begin{cases} 1 & \text{if } I(i; j) == k \\ 0 & \text{otherwise} \end{cases}$$

4. Iteratively 'k'(<B) such layers are obtained.

Each of these 'k' layers can now be considered as an independent image.

Thus after explosion a pixel with value '1' in a layer 'l' can be interpreted as, "there is a pixel of value 'l' at that location in the original image".

Smoothing:

1. Smoothing refers to the process of convolving an image function with an appropriate gaussian function, to lessen the irregularities or sharpness in an image.

2. 'Sigma' value of the gaussian function is certainly unknown and hence a finite set of possible values is maintained.

3. Each time matching is done for all possible values of 'sigma' starting from bigger values (coarse fit) to smaller values (finer fit) to align the model to target

4. Smoothing is done both along an image and across the image layers
   - df1 = df * h(x;y)
   - df2 = df1 * h(z)

Thus after smoothing a pixel with a non-zero value in a layer 'l' can be interpreted as, "there is a pixel of value 'l' somewhere near that location in the original image."

Why are they needed?

A very memory intensive data structure of a Distribution field undoubtedly goes against the necessity of its use, but the positive's that works in its favor certainly overcasts that.

To mention, it is through this splitting of image feature information into multiple levels is that we are able to smoothen an image without loss of much information as it would have been the case if the image was directly smoothen.

Smoothing an image before matching it for the best possible location of the desired area to be tracked certainly produces much better results as compared to matching directly with an image having all the features intact "as it is"; because in the later case even a small gradient change would cause the tracker to loose the object in that image frame, however such a change is hardly noticeable if a diffused image is used for matching as is in the former case.

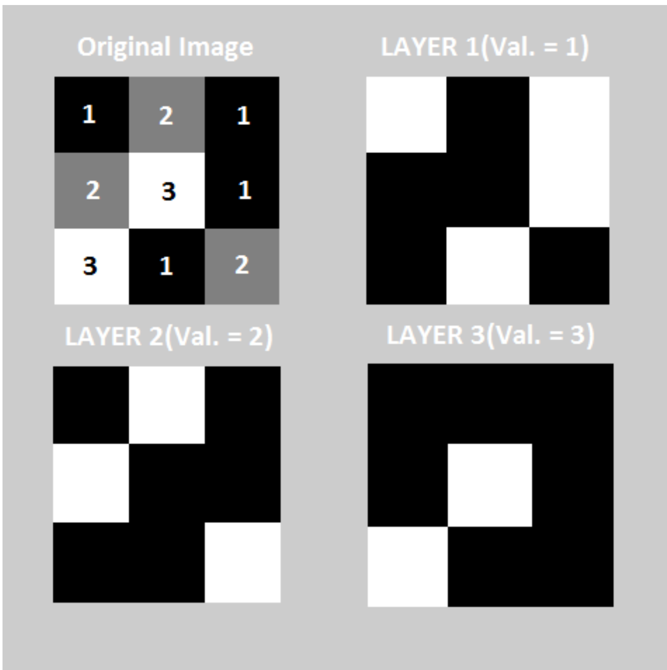


Figure 1 Explosion

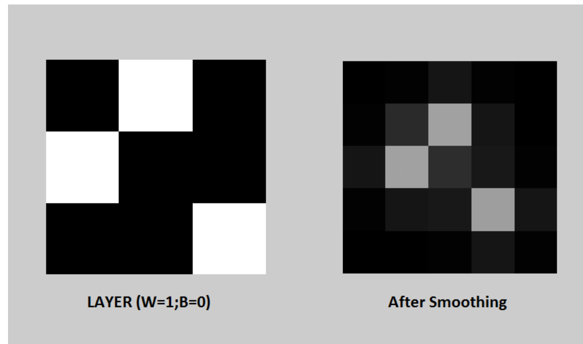
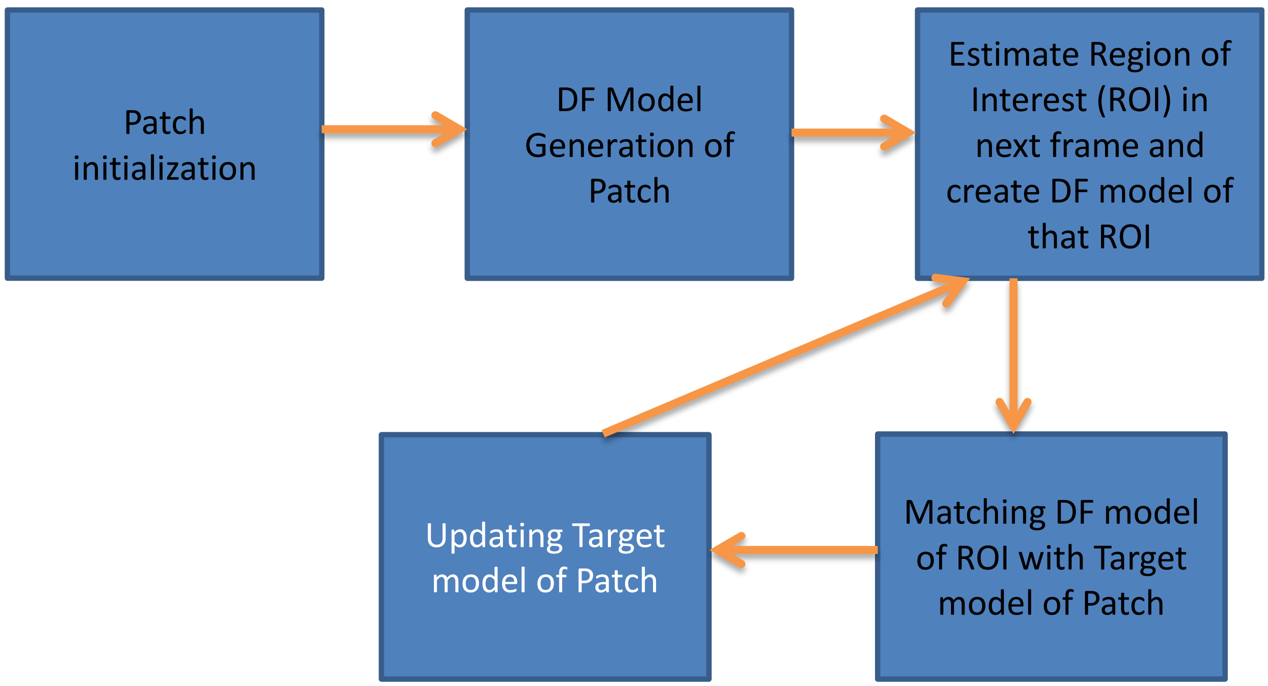


Figure 2 Smoothing

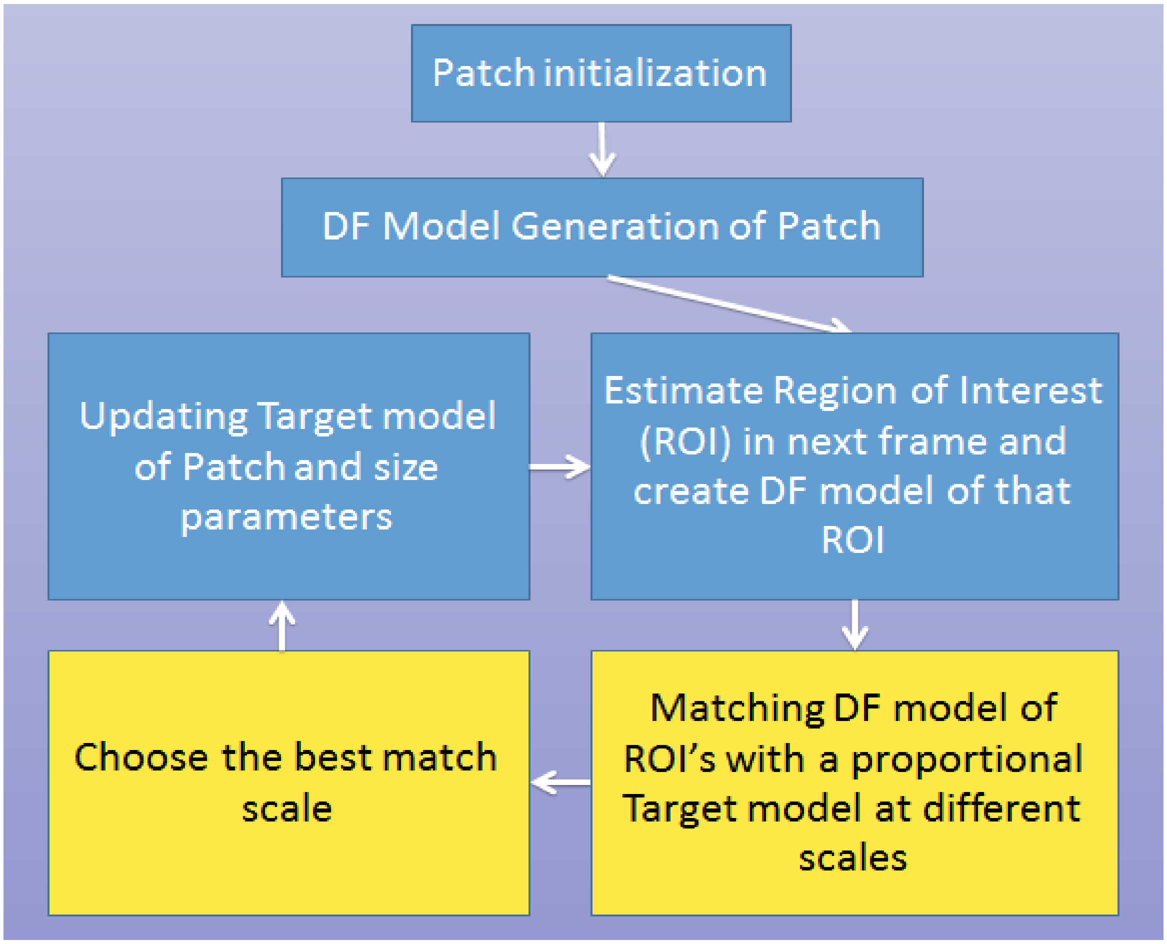
Frame work of original algorithm

-Window initialization across the object of interest in First Frame
- Exploding: A single image is split into n number of layers where each layer contains similar intensity pixels (n is the number of intensity bins)



- Smoothing: A Gaussian kernel is used to smooth in each layer while another kernel is used to smooth across layers. Now intensity in each layer represents the probability of finding a particular intensity near that pixel
- An estimation of ROI is made based on a motion model that uses previous frame motion vector, object window and kernel size.
- Implemented using Gradient descent and amount of matching is calculated using L1 Distance (L1 Distance is sum of absolute differences among the source and target image pixels)
- Target model is updated by adding a weighted component of current DF with previous model
- Updating the initialization point and motion vector
- Same steps are iterated for all other upcoming frames

Modified algorithm



We have added scale invariance and depth estimation features to this algorithm.

Scale Invariance:

Target window size gets regularly updates to tightly bound the targeted object even when it changes its scale

Our Approach-

We scale the target model to different scales and find their match with ROI's of different sizes and choose the best matching ROI

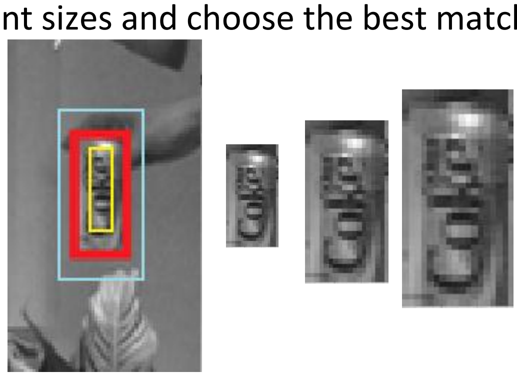


Figure: The image on left shows the image cropped on basis of a motion model. The different rectangles represent the different sizes of ROI to be matched with target model at different scales (represented by cans)

Results

For the purpose of comparing the original algorithm with our modified version we created our own dataset to introduce change of scale.

Result of **tracking with original algorithm-**

Frame1

Frame 100

Frame 134

Result of **tracking with modified algorithm-**

Frame1

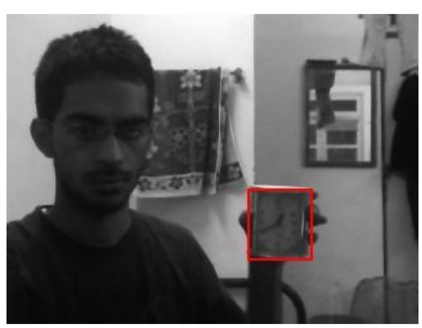
Frame 100

Frame 134

With another dataset

Frame1

Frame 144



Frame 435

References

- **Distribution Fields** –Laura Sevilla-Lara, Erik Learned-Miller 2011
- Laura Sevilla-Lara and Erik Learned-Miller.
  Distribution Fields for Tracking
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

Acknowledgments

We are very thankful to Vidit Jain (scientist in Yahoo! Labs and IITK alumnus) from whom we came to know about recent developments in tracking and work in Distribution Fields. We are also grateful to Prof. Amitabha Mukerjee for the valuable time he devoted to guide us throughout this project.