Distribution Fields for Tracking

Laura Sevilla-Lara Erik Learned-Miller
University of Massachusetts Amherst
{lsevilla, elm}@cs.umass.edu

Abstract

Visual tracking of general objects often relies on the assumption that gradient descent of the alignment function will reach the global optimum. A common technique to smooth the objective function is to blur the image. However, blurring the image destroys image information, which can cause the target to be lost. To address this problem we introduce a method for building an image descriptor using distribution fields *(DFs), a representation that allows smoothing the objective function without destroying information about pixel values. We present experimental evidence on the superiority of the width of the basin of attraction around the global optimum of DFs over other descriptors. DFs also allow the representation of uncertainty about the tracked object. This helps in disregarding outliers during tracking (like occlusions or small misalignments) without modeling them explicitly. Finally, this provides a convenient way to aggregate the observations of the object through time and maintain an updated model. We present a simple tracking algorithm that uses DFs and obtains state-of-the-art results on standard benchmarks.*

1. Introduction

In this paper, we address the problem of searching for a target in an image using gradient descent methods, i.e. local methods of searching for a target match.

To implement tracking using local search, we must choose a representation for the target and the patch to which we are comparing it. There is a fundamental tension between two conflicting goals when choosing such an image representation. On the one hand, we would like a matching function whose global optimum represents the true position of the target rather than a spurious match. We refer to this property as the *specificity* of the descriptor.

On the other hand, we would like the optimization landscape (of the matching function) to have few local minima. We refer to this as the *landscape smoothness* criterion.

There are many descriptors that exhibit one property but not the other. For example, by blurring an image patch representation, we can produce a representation whose optimization landscape is quite smooth. However, the specificity of such a blurred image descriptor has been compromised since the blurred image patch has lost significant information about the original target to be tracked.

In this paper, we use a representation for targets and images that has not previously been used in the tracking literature. We refer to this representation as *distribution fields* (DFs).[1] While DFs have been used in a variety of applications in computer vision, the particular technique we use to build a DF representation of a target is novel. We show that our DF descriptor is extraordinarily good at satisfying the

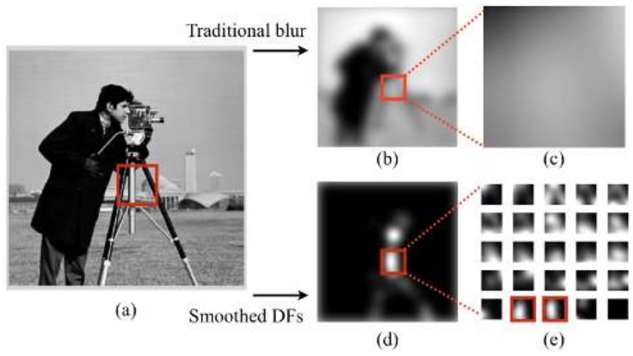

Figure 1. **Information preserved using smoothing on a DF. (a)** Original image. **(b)** Image smoothed with traditional blur. **(c)** Patch of image (b) where there central bar used to be. **(d)** Layer of the DF corresponding to the intensity value of the bar. **(e)** Collection of patches of the DF under the location of the central bar. When an image is blurred, the new pixel values are a combination of the neighboring pixels around them, and all the information is collapsed into a single number. In this case, for example, the central bar of the tripod has blended in completely with the background (c). When an image is exploded into a DF and smoothed, different values are blurred separately in different layers, and therefore the pixel values are not combined. In this case, the cameraman has been exploded into a DF with 25 layers, and blurred with the same kernel that was used for smoothing with the traditional blur. The central bar of the tripod is still represented in one of the layers.

[1]Portions of this work have been described in a technical report [22].

specificity and smooth landscape requirements of a good tracking algorithm. We present several types of evidence supporting this claim.

First, we show that our DF representation for tracking has a wider *basin of attraction* around a target's location than a large number of other descriptors that have been used in the tracking literature (see Section 5). Second, we show that a simple tracker built from this descriptor outperforms all other trackers on a standard tracking data set. Finally, we show that our tracker does not drift in a very long video sequence.

The paper is organized as follows. In Section 2 we describe the previous work on descriptors for tracking. In Section 3 we define DFs and the operators over them. In Section 4 we describe the tracking algorithm. Section 5 shows experimental evidence on the superiority on the basin of attraction and tracking performance of DFs. Section 6 closes with a discussion.

2. Previous work

Tracking algorithms have different aspects including motion modeling, image representation and model update. The main focus of this work is the representation of the image using a descriptor that can overcome the challenges of visual tracking.

One common approach is to use a template to represent the object. This template can consist of the intensity values, gradient information, or other features [3]. These techniques have limitations because they might be overly sensitive to the spatial structure of the object. This means that even if the optimization reaches the global optimum, this might not coincide with the correct position of the object due to changes in appearance. Robust metrics [18] alleviate this problem, but performance decays in long sequences [14]. DFs allow the representation of uncertainty in the descriptor, where small misalignments or occlusions can be represented as "unlikely" events as opposed to "impossible" events, mitigating the oversensitivity to spatial structure.

Another problem with template-based methods is that the objective function might not be smooth enough to reach the global optimum, as pointed out by Szeliski [24]. Often, the function is smoothed by blurring the image, for example using a Gaussian pyramid [1]. Recently, it has been proven that the Gaussian pyramid is not always the best choice for smoothing the objective function [20], and alternative blur kernels have been derived [20] specifically for smoothing the optimization landscape. Blurring the image has the undesired effect of destroying information about the pixel values. Depending on the size of the target, the levels of the pyramid and the characteristics of the background, the target might disappear completely. In the DF framework, the layered, or channel-by-channel, blurring technique allows smoothing the objective function without the mix of information that occurs in traditional blurring. This process is illustrated in Figure 1. The result is a smooth function with a wider basin of attraction around the object location than other descriptors. Figure 5 shows an example in one of the benchmark frames.

An alternative to building a template is using a histogram to describe the object [8, 6]. Histogram-based (also called kernel-based) descriptors integrate information over a large patch of the image. As a result they are not overly sensitive to spatial structure and they provide a larger basin of attraction. These methods have had a lot of success because they are fast, simple, and invariant to many pose changes. The main problem of kernel-based methods is the loss of spatial information that happens when building the histogram. This creates ambiguity in the optimization function [13] decreasing the specificity of the descriptor. In order to resolve these ambiguities, the size of the descriptor should be expanded. Recent efforts include some spatial information in the descriptor by using multiple kernels [13], or multiple patches [11]. These methods improve the performance of the single histogram descriptor, but require other mechanisms to decide the number and shape of the kernels. Fan et al. [12] propose a solution for the placement of the kernels, but a change in object appearance may make these kernels unstable. Also, if the number of these kernels is small, the tracking accuracy is more vulnerable to occlusion or abrupt motion. Other additions are higher order statistics [4] or temporal information [5], and using feature selection [7, 16, 26]. DFs capture the rich and robust information contained in a histogram while preserving the spatial structure of the object by having a distribution at each pixel and can be viewed as a soft combination of template-based and histogram-based descriptors.

DFs are a generalization of many previous image representations used for different purposes. These previous descriptors can be viewed as special cases of DFs, and they have many of the desired properties listed above. To our knowledge, only the general case of DFs together with the set of operators described in Section 3 presents all the properties.

In background subtraction, Elgammal et al. [10], and Stauffer and Grimson [23] use DFs for modeling the background. A pixel is classified as background or foreground depending on the probability of belonging to the background. These descriptors can adapt to changes in appearance and be robust to certain noise and illumination. However, these descriptors do not spread information in space.

In object detection and recognition, descriptors like HOG [9] and SIFT [17] use histograms of gradients. These can be viewed as downsampled DFs with gradient as the feature space. The large "support" of each histogram yields a smooth descriptor and a smooth objective function. However, since the size of this "support" is fixed, the basin of

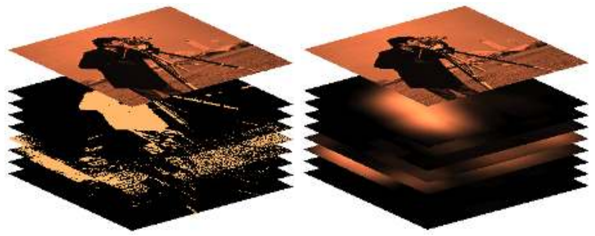

Figure 2. **Left**. DF after exploding the "cameraman" image. (The original image is shown superimposed on the DF for clarity.) The number of brightness levels (or layers) has been quantized to 8. **Right.** The same DF after smoothing in the dimensions of the original image.

attraction is much more limited.

Another use of DFs was demonstrated by Learned-Miller [15] in developing the congealing framework for joint image alignment. In this work, the likelihood of each image is maximized with respect to the DF defined by the set of images. Congealing [15] has a large basin of attraction for alignment, since a collection of images can smooth the optimization landscape.

Finally, the bilateral filter [25] introduced a way of smoothing an image such that both proximity in space and feature value are taken into account to preserve image detail, which is similar to blurring using DFs.

3. Description of Distribution Fields

A DF is an array of probability distributions, one at each pixel of the "field". This distribution defines the probability of a pixel of taking each feature value. For example, if the feature space is gray-scale intensity, then at each pixel there is a probability distribution over the values 0-255.

Representation: A DF is represented as a matrix d with $(2+N)$ dimensions, where the first two dimensions are the width and height of the image, and the other N dimensions index the feature space that we choose. For example, if the feature space is intensity, then an image of size $m \times n$ yields a 3D DF of size $m \times n \times b$, where b is the number of intensity feature values, or bins. For a higher dimensional feature space, such as gradient, we can build a 2D distribution at each pixel location, yielding a DF of four dimensions.

Construction: *Exploding an image* into a DF results in a Kronecker delta function at each pixel location. In particular, exploding an image I into d with as many bins as features values is defined by

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

where i and j index the row and column of the image, and k indexes the possible values of the pixel. We call the collection of bins at a fixed depth k a $layer$. This produces a probability distribution at each pixel since the sum of the components of each column is 1. The left side of Figure 2 shows the results of computing this DF for the well-known "cameraman" image.

At this point, the DF representation contains exactly the same information as the original representation, albeit in a larger representation. We now show how to "spread" the information in the image without destroying the brightness values as occurs with traditional blurring.

The right side of Figure 2 shows a smoothed version of the DF on the left. The 3D DF has simply been convolved with a 2D Gaussian filter which spreads out in the x and y dimensions, but not in the feature dimension. That is, each layer k of the smoothed DF d_s is computed as

$$d_s(k) = d(k) * h_{\sigma_s}, \tag{1}$$

where h is a 2D Gaussian kernel of standard deviation σ_s, and $*$ is the convolution operator.

Prior to convolution, we could interpret any value of 1 in layer L of a DF to mean "there is a pixel of value L at this location in the original image." After convolution, the semantics of the smoothed DF is, for any non-zero value in a layer L, "there is a pixel of value L somewhere near this location in the original image." Thus, the convolution has introduced positional uncertainty into the representation. *A critical point is that no information has been lost about the* **value** *of pixels in the original image, only about their position.* This is because there has been no mixing of pixel values during the convolution process.

It is easy to show that if the convolution kernel in Equation 1 is itself a probability distribution, then the smoothed d_s maintains the property that each column of pixels integrates to 1,[2] and hence is still properly called a DF.

The previous discussion describes smoothing a DF in the x and y image dimensions. Smoothing can also take place in feature space. This allows the model to explain small changes due to subpixel motion, shadows, and changes in brightness. In a grayscale image, this smoothing is a 1D Gaussian filter over the third dimension. Each of the columns of d_s can be smoothed to produce d_{ss} as

$$d_{ss}(i,j) = d_s(i,j) * h_{\sigma_f}, \tag{2}$$

where h_z is a 1D Gaussian kernel of standard deviation σ_f.

In summary, exploding an image into a DF and smoothing it can be viewed as introducing uncertainty about the object appearance. A DF is then a compact representation of the image itself and a set of its "neighboring" images. These images are the result of transforming the original image with small changes in appearance and in location.

[2]This property breaks down at the boundaries of images. In order to avoid this problem, the missing information outside the boundaries is filled with uniform distributions.

These are weighted according to the simple assumption that the most likely event is that the image will stay the same, and larger changes are less likely.

Comparison: The comparison between DFs that different images yield can be done with any distance function. In this paper we use the L_1 distance between the two arrays d_1 and d_2 as:

$$L_1(d_1, d_2) = \sum_{i,j,k} |d_1(i,j,k) - d_2(i,j,k)|. \tag{3}$$

Combination: Combining the information of several DFs can also be useful. In tracking we combine the DF of initial model and the DFs of new observations using a component-wise convex combination of them, which also yields a DF:

$$d_{t+1}(i,j,k) = \lambda d_t(i,j,k) + (1-\lambda) d_{t-1}(i,j,k) \tag{4}$$

By combining DFs of different instances of the same object we build a non-parametric data-driven model of the distribution at each pixel. This is useful for learning the statistics of the appearance of the object during tracking.

4. Tracking algorithm details

DFs can be used in a simple tracking algorithm. A model of the target is created by exploding the image that contains the target into a DF and smoothing it. Searching for the target in a new frame consists of building a new DF by also exploding and smoothing the new frame, and following the direction where the gradient of the L_1 difference between the DF of the model and the underlying part of the bigger DF representing the new frame descends. Once a local minimum is reached, the model of the target is updated, using a linear combination of the model and the new observation, as in Equation 4.

For better performance, we use a hierarchical approach. Instead of using a single DF to represent the target, we use a small set of DFs, where each of them is built using an increasing value of the parameter σ_s, which regulates the amount of spatial blur. These DFs contain information at different frequencies. At each frame, we use a coarse-to-fine strategy. The most smoothed DF is used to start the search, until it reaches a local minimum. This position is the start for the search in the second DF.

The method for choosing the value of the parameters λ and σ_f is using leave-one-out cross validation. The result of picking σ_f separately for each video happens to coincide for all of them to be $\sigma_f = 10$. This is also the case for $\lambda = 0.95$.

Parameter b corresponding to the number of bins was chosen, for speed, as the smallest power of two that does not hurt the performance of the videos. This is $b = 16$.

Algorithm 1 Tracking with distribution fields

```
Input: $V$ = video sequence.
    $I$ = patch containing target in frame 1.
    $\boldsymbol{\sigma}_s$ = set of spatial smoothing parameters.
    $\sigma_f$ = brightness smoothing parameter .
    $b$ = number of brightness bins ($b = 16$).
    $\lambda$ = mixing parameter ($\lambda = 0.95$).
Output: $(x,y)_f$ {Positions of target at each video frame $f$ in $V$}
 1: Initialize $d^i_{model} = explode(I) * h_{s(i)} * h_f, i \in 1,..|\boldsymbol{\sigma}_s|$
 2: Initialize target location $(x,y)$ to center of patch $I$.
 3: for $f = 2 \rightarrow |V|$ do
 4:   for $i = 1 \rightarrow |\boldsymbol{\sigma}_s|$ do
 5:     $d^i_f = explode(f) * h_{s(i)} * h_f$
 6:     $(x', y') = \underset{(x,y)}{\text{argmin}}\, L_1(d^i_f(x,y), d^i_{model})$
 7:     $(x,y) = (x', y')$
 8:   end for
 9:   $d_{model} = \lambda d_{model} + (1-\lambda) d_f(x,y)$
10: end for
```

The schedule of σ_s for each video was chosen also using cross validation but conditioned on the size of the target. For each video, we choose the schedule of σ_s that performs best for the video whose target is closest in size. The impact of all parameters is studied in Section 5.3. The only motion model present in the algorithm is the assumption of constant velocity for computing the start of the search.

We now summarize the procedure in pseudo-code.

Here h_s and h_f are a 2D and a 1D Gaussian filters built with σ_s and σ_f respectively.

The computational cost of our algorithm is dependent on the number of bins used. We used $b = 16$. In a naive implementation, the running time is then 16 times that of template-matching with a Gaussian pyramid. However, the step that requires extra computation is the convolution of each layer of the DF. It is important to notice that this step can be completely parallelized. We have implemented this algorithm in a GPU in real-time at 70 frames per second on a PAL video (768×576 pixels).

5. Experiments

One of the main advantages of DFs is the width of the capture range around the position of the target. To illustrate this, we first compare the width of the basin of attraction of DFs to other descriptors. Second, we show that the tracking algorithm that uses DFs is able to outperform other state-of-the-art methods in standard benchmarks.

5.1. Experiments on basin of attraction of Distribution Fields

Here we evaluate the improvement in the basin of attraction achieved by using DFs compared to other descriptors.

Experiment description: For an objective function $f(x, y)$ and a point p, the *basin of attraction* is the region around the point p from which descending the gradient of $f(x, y)$ leads to p. The size of this region is crucial for tracking algorithms that follow gradient descent to avoid exhaustive search. In this experiment, given an image, a patch is randomly selected and displaced in the horizontal direction. The task is to find, using gradient descent, the true position of the patch. This procedure is illustrated in Figure 3. Since there is no noise or distortion, this task isolates the problem of creating a spatially smooth objective function to that of creating an objective function robust to changes in appearance. 289 images from the UWA data set[3] are used. The size of the patches is 30×30, and they were displaced from 1 to 30 pixels in each direction of the horizontal axis. We compare DFs to six other related or commonly used descriptors. These are the three traditional techniques: sum of squared distances (ssd), normalized crossed correlation (ncc), sum of squared distances of the blurred image (blur) and the three related descriptors meanshift using Bhattacharyya distance (ms-bhatt), meanshift using L_1 (ms) and multiple kernel tracking descriptor (mkt). We use both L_1 (df) and Bhattacharyya (df-bhatt) for the DF descriptor. The size of the kernel used for blurring is the same for all different descriptors that use blur. Figure 3 shows an example of the result of evaluating each objective function around the true location of a patch for one particular image.

Results: Figure 3 is the cumulative histogram of the size of the basin of attraction for each image. It shows how often each descriptor is able to successfully follow a gradient back to its original position as a function of the original displacement. For example, the DF descriptor (cyan and magenta lines) is able to follow a gradient back to its original position from a displacement of 15 pixels more than 90% of the time, while normalized cross correlation can only do this about 20% of the time.

There are two important points to take from this graph. The first is that the basin of attraction of DFs is larger than for any other descriptor. The reason for the superiority over traditional blur is that blurring the DF descriptor, as described in Equation 1, does not mix the values of different pixels. The reason for the superiority over the other kernel-based descriptors is that the size of the signature is increased and therefore there is more specificity among patches. Kernel-based descriptors are a special case of DF, where the value of σ is fixed and only the distribution of the

[3]http://www.cs.washington.edu/research/imagedatabase/groundtruth/

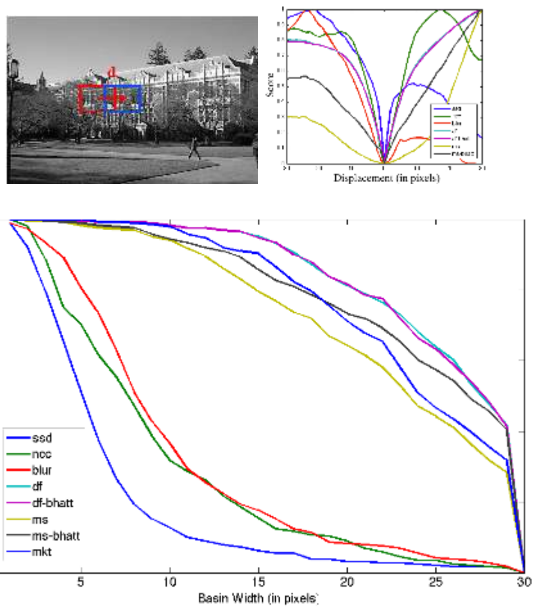

Figure 3. **Top left**. Experiment to evaluate width of basin of attraction. The dashed rectangle is the true position of the patch, and it is displaced by a distance $d = 30$. The basin of attraction tests show how often a patch is able to find its original position at this displacement. **Top right**. One instance of the different distance metrics evaluated translating a patch 1-30 pixels in both directions horizontally. **Bottom**. This plot shows, for a variety of descriptors, the cumulative distribution of basin of attractions.

center pixel is used. The second point is that the basin of attraction of a DF is consistently superior despite the distance metric used, this is, L_1 and Bhattacharyya. Thus the descriptor in this experiment is more influential than the distance metric.

5.2. Comparative tracking experiments

Data set: To evaluate the performance of the tracking algorithm we use the list of videos compiled by Babenko et al. [2] that is publicly available. [4] Since there is no standard benchmark for tracking, we choose these videos because they are the ones that most authors have compared to and they represent a valuable *de facto* standard for evaluation. They exhibit a wide range of phenomena from occlusion, object deformation, significant change in object appearance (subject turns 360 degrees out of plane), moving complex backgrounds (surfing). They also show a variety of objects being tracked such as faces, toys, and soda cans.

Algorithms for comparison: We compare the performance of our algorithm to three other algorithms. We use the trackers that, to the best of our knowledge, perform best

[4]The videos can be found at http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml One of the links to the videos was broken.

Video	DF	PROST	MIL	MKT
tiger1	**88.57**	79	43.14	12.86
david	**100.00**	80	58.91	30.43
sylvester	66.79	**74**	73.88	20.90
girl	73.00	**89**	55.20	7.00
faceocc	**100.00**	**100**	77.28	6.21
faceocc2	**98.76**	82	78.13	27.33
coke11	**75.86**	-	17.93	27.59
surfer	**94.67**	-	56.00	62.67
dollar	**100.00**	-	90.76	15.38
tiger2	**81.94**	-	46.39	8.33
cliffbar	**87.69**	-	72.31	-

Table 1. **Percentage of correctly tracked frames**

Video	DF	PROST	MIL	MKT
tiger1	**6.49**	7.20	17.60	79.13
david	**9.97**	15.30	23.45	98.62
sylvester	15.92	**10.60**	10.62	49.24
girl	21.57	**19.00**	32.76	105.05
faceocc	**5.00**	7.00	27.28	102.47
faceocc2	**11.25**	17.20	21.06	87.68
coke11	**7.19**	-	20.85	20.33
surfer	**5.20**	-	12.06	17.54
dollar	**5.26**	-	15.15	81.26
tiger2	**6.75**	-	18.97	69.48
cliffbar	**7.77**	-	12.23	-

Table 2. **Mean distance to the ground truth**

on most videos of this data set: MIL [2] and PROST [21]. MIL uses unsupervised online learning over Haar features to train a discriminative model over the two classes background and foreground. In PROST, three different trackers are used and combined using a cascade. PROST has only been run on a subset of six of the videos, and therefore we can only compare directly to it on these. In addition, our tracker is most similar in spirit to the Multiple Kernel Tracker [13] (MKT), and so we implemented and ran it on the same suite. The kernels used in our implementation are those described in their experiments, which are an Epanechnikov kernel and a roof kernel. The MKT descriptor is updated with the same scheme as used for DFs.

Quantitative analysis: We use two different metrics for the analysis of the tracking results. These are the mean distance to the ground truth (Table 2) and the percentage of frames correctly tracked. A frame is correctly tracked if the track and the ground truth have an overlap that is larger than the union of their areas. This is, if the track is rectangle A, and the ground truth is rectangle B, then a frame is correctly tracked if $(A \cap B)/(A \cup B) > 0.5$. This is shown in Table 1. We consider this metric to be much more informative than the distance, since once the track is lost the distance to the ground truth is somewhat arbitrary, and might bias the average distance. However, we report both for consistency with existing literature. Since MIL is stochastic, we show the average of the five runs that they report.

The tables show superiority of DF with respect to MIL and MKT in all 11 videos. Compared to PROST, DF is better in 4 out of the 6 videos that they use for comparison. In the video "faceocc" where both algorithms track correctly 100% of the frames, the accuracy of DF is better.

Qualitative analysis: The tracking algorithm is able to overcome different challenges like moderate occlusion and moderate changes in appearances due to pose and illumination change. Occlusions can cause drift in systems that are updated online without supervision. When the DF that represents the model of the target is updated with an occluder, this is represented in the distributions with small weight, as an unlikely event. If the occluder is not present during many frames, the model DF will not be polluted with the occluder, and will successfully avoid drift. This is the case of many of the videos of the benchmark, as depicted in Figure 4. This is also the case for other changes in appearance, like changes in illumination and pose. In the two videos where DFs perform worse ("girl" and "sylvester") the track drifts due to very drastic changes in appearance. In the first one, the girl whose face is being tracked turns around twice before the track drifts. In the second, the object undertakes large pose and illumination changes simultaneously.

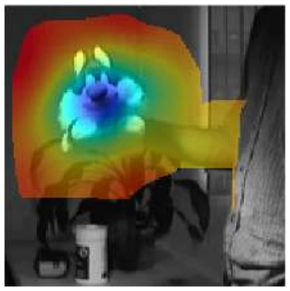
Figure 5. **Basin of attraction**. If the center of the target is located at any of the colored pixels, gradient descent leads to the correct position (within a 3×3 neigborhood). These pixels are color-coded according to the value of the objective function at that point.

5.3. Parameter Analysis

All parameters in the algorithm are chosen using cross validation. In this section we further study the effect of each of them in tracking performance. The most important parameters are σ_s and σ_f, which are the standard deviation of the Gaussian filters in image space and feature space respectively. In the algorithm we use a coarse-to-fine strategy for σ_s, where we convolve the DF first with a large, flatter filter, and we progressively sharpen it. The initial large filters smooth the optimization landscape, allowing the recovery of long displacements. The final smaller filters increase the specificity of the descriptor, allowing the optimization function to be more discriminative. If the value

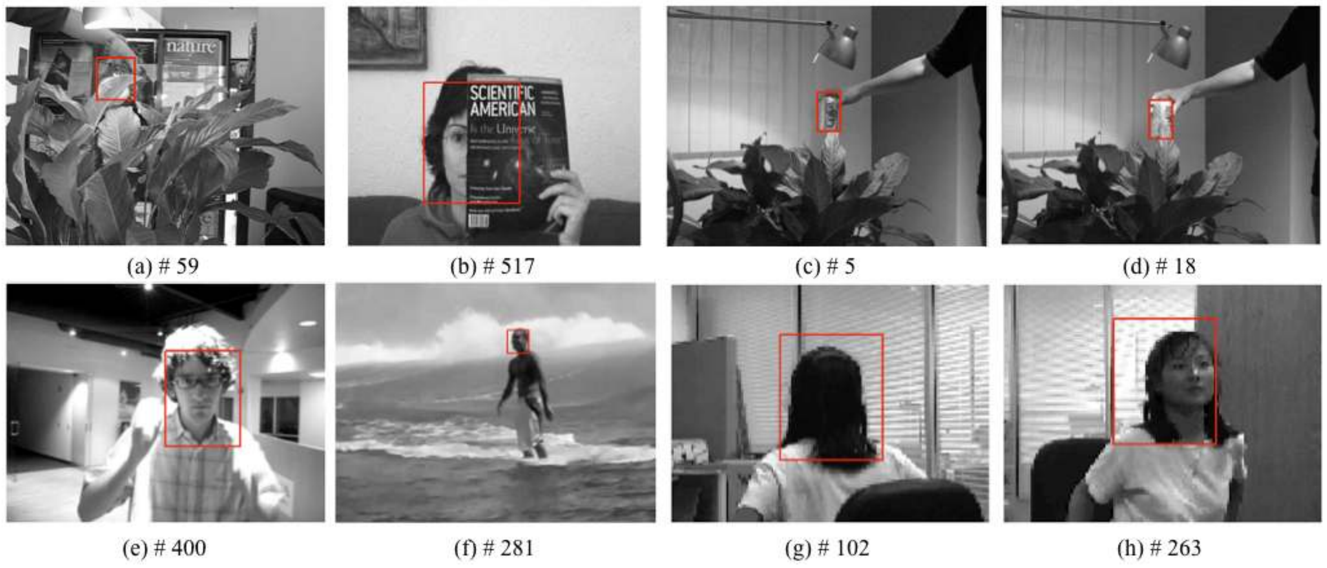

Figure 4. **Sample Frames.** Our algorithm overcomes limited occlusions (a, b), moderate changes in illumination (c, d, e), and it's robust to low resolution outdoors sequences (f). Drift occurs when the changes in appearance are prolonged and very drastic (g, h).

of σ_s is too large, all patches might be too similar, and the track might be lost. If it is too small, the tracker might get stuck in a local minimum. Figure 6 shows the result of running the tracking algorithm with different configurations of σ_s. The bars for a given video show the sizes of σ_s in ascending order. Although improvements in accuracy are not very smooth (since a single occlusion can cause the track to be lost forever), often larger targets are better tracked with larger values of σ_s. For example, "girl" and "faceocc" seem to perform better with larger sizes of σ_s. This seems reasonable, because descriptors that represent small patches can become indistinguishable from each other more easily when blurred, since they have lower dimensionality. This suggests that the parameter is somewhat consistent and what is most important, is generalizable. That is, the best value for a particular target will be very good for a different target in a different video if they have similar sizes.

The case of the value of σ_f in feature space is similar to σ_s in space, and in general smaller targets are better tracked with smaller kernels. However, there are other factors that also influence the best value of σ_f, such as the similarity between background and foreground and the variance in the appearance of the model. If σ_f is very small, large targets tend to perform worse and videos with small targets tend to perform better. If σ_f is very large, the opposite is true. This explains the large variation of the performance in small and large values of the parameter in Figure 7. Potentially, both σ_s and σ_f could be learned to be adaptative for the particular characteristics of the video, but in this case the suite of videos is too reduced and diverse.

Parameter λ controls the rate at which the model is updated as described in Equation 4. If the model is updated very fast, small errors accumulate quickly and cause the track to drift. This trade-off is shown in Figure 7. If the model is updated very slowly, it may become outdated since it may be unable to reflect changes in the appearance.

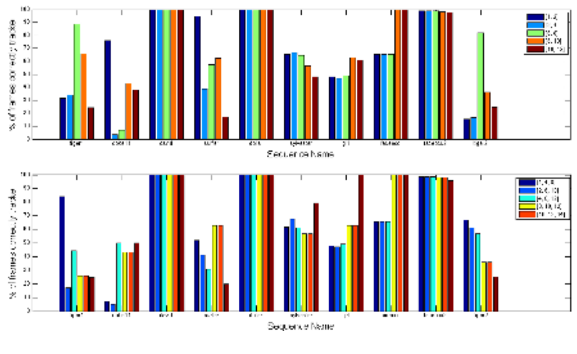

Figure 6. **Percent of frames correctly tracked across different σ configurations**. **Top.** Using a two-level pyramid. **Bottom.** Using a three-level pyramid.

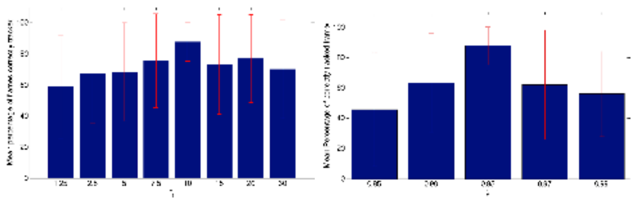

Figure 7. **Mean percentage of frames correctly tracked vs. value of σ_f and λ.**

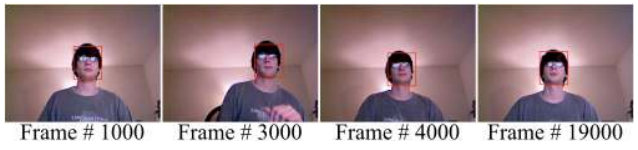

Figure 8. **Sample frames of a sequence with 19000 frames.** Although in some frames (3000 to 4000) there is a certain amount of background included in the model, the tracker successfully recovers the correct position of the head.

5.4. Drift avoidance over long sequences

Tracking algorithms that are updated online tend to drift over long sequences [19]. Once the model starts including a part of the background, errors will accumulate and the track will drift away from the target. We argue that our algorithm for tracking with DFs is able to avoid this problem naturally by keeping a model of the target that is flexible enough to account for changes in appearance but allows a certain memory on the appearance model. We ran our algorithm with the parameters chosen as in Section 4 in a sequence of 19000 frames. Although during some frames there is some part of the background included (frames 3000 - 4000), the tracker successfully recovers as shown in Figure 8.

6. Conclusion

In this paper we have used a descriptor called DF for the tracking of general objects in video sequences. Tracking with DFs has two contributions. First, they have a larger basin of attraction than other similar descriptors, which prevents the search from getting stuck in local minima. Second, DFs present a variety of advantages for tracking, they include spatial information in the kernel-based framework, which resolves ambiguity and overcomes the undersensitivity to spatial structure. They also resolve the oversensitivity that other descriptors have to the geometric structure of the target, and they are able to model slow changes in appearance and pose and be robust to minor occlusions.

We believe that DFs are a fertile framework for image comparison and there are many improvements to our algorithm that could be explored, such as modeling occlusion, combining information over multiple feature spaces or including a memory model to store different poses.

Acknowledgements

This work was supported in part by National Science Foundation under CAREER award IIS-0546666.

References

[1] S. Avidan. Support vector tracking. In *PAMI*, 2001. 2

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, 2009. 5, 6

[3] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *IJCV*, 2004. 2

[4] S. T. Birchfield and S. Rangarajan. Spatiograms versus histograms for region-based tracking. In *CVPR*, 2005. 2

[5] K. J. Cannons, J. M. Gryn, and R. P. Wildes. Visual tracking using a pixelwise spatiotemporal oriented energy representation. In *EECV*, 2010. 2

[6] R. T. Collins. Mean-shift blob tracking through scale space. In *CVPR*, 2003. 2

[7] R. T. Collins and Y. Liu. On-line selection of discriminative tracking features. In *ICCV*, 2003. 2

[8] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000. 2

[9] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[10] A. M. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. In *EECV*, 2000. 2

[11] Z. Fan, M. Yang, and Y. Wu. Multiple collaborative kernel tracking. In *CVPR*, 2005. 2

[12] Z. Fan, M. Yang, Y. Wu, G. Hua, and T. Yu. Efficient optimal kernel placement for reliable visual tracking. In *CVPR*, 2006. 2

[13] G. Hager, , G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. In *CVPR*, 2004. 2, 6

[14] A. D. Jepson, D. J. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. In *CVPR*, 2001. 2

[15] E. G. Learned-Miller. Data driven image models through continuous joint alignment. *PAMI*, page 2006, 2006. 3

[16] A. P. Leung and S. Gong. Mean shift tracking with random sampling. In *BMVC*, 2006. 2

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2

[18] H. N. Marcel, M. Worring, and R. V. D. Boomgaard. Occlusion robust adaptive template tracking. In *ICCV*, 2001. 2

[19] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. In *BMVC*, 2003. 8

[20] H. Mobahi, C. L. Zitnick, and Y. Ma. Seeing through the blur. In *CVPR*, 2012. 2

[21] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In *CVPR*, 2010. 6

[22] L. Sevilla-Lara and E. Learned-Miller. Distribution fields. Technical report, University of Massachusetts Amherst, 2011. Supplied as additional material `techreport.pdf`. 1

[23] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *PAMI*, 2000. 2

[24] R. Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2:1–104, January 2006. 2

[25] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998. 3

[26] Z. Yin, F. Porikli, and R. T. Collins. Likelihood map fusion for visual object tracking. In *IEEE Workshop on Applications of Computer Vision*, 2008. 2