Distribution Fields

Laura Sevilla-Lara, Erik Learned-Miller

Technical Report UM-CS-2011-027, University of Massachusetts Amherst

1 Introduction

Optical flow, tracking, background subtraction, stereovision, alignment, registration have in common that they try to find correspondences between parts of the images. These correspondences are often computed by mapping the images to descriptors and comparing these. A good descriptor that enables robust matching will be therefore useful for all of these low-level problems. Some of the properties that such descriptor should have are: robustness (to image noise, small misalignments), smoothness to degradations (to spatial location, scale, brightness), smoothness in its representation and ability to adapt to different images easily.

Having a common representation for multiple problems allows the combination of different problems to help constrain the solution better. Many examples of recent work leverage the combination of tracking and optical flow [4], tracking and background subtraction [22, 21] or segmentation and optical flow [20].

In this document we introduce *distribution fields* (DFs) as an image descriptor for low-level vision problems. We argue that many current image descriptors (i.e.: SIFT, HOG, kernel-based descriptors, geometric blur etc.) can be explained in terms of a DF, which is a distribution at each pixel over the feature space. This way different descriptors can be compared and combined. Many of these descriptors have some of the important properties described above. However, to our knowledge, only DF together with a set of operations associated to them can capture all of them.

The rest of this document is organized as follows. In section 2 we explain the relationship between DFs and other descriptors. In section 3 we describe DFs, the operations over them and the meaning of these operations. In section 4 we present experiments comparing the basin of attraction of DFs and other descriptors'. In section 5 we apply DFs to object tracking and show state-of-the-art results.

2 Related work

The idea of using a distribution per pixel over the feature values has been used successfully for several vision problems. Stauffer and Grimson [22] used a parametric distribution at each pixel for tracking through background subtraction. In this work, there is a foreground model that is a distribution field. Each distribution is modeled as a mixture of Gaussians. A pixel is classified as foreground or background based on the probability density of the pixel value. This descriptor is robust to certain noise, and has smooth degradations to brightness. However, the parametric model has certain limitations as it is explained in the work of Elgammal et al. [11], like the flexibility of the model to adapt to changes. In this work, also a DF is used for background subtraction, but the background model is built nonparametrically by treating each frame as a sample. As it is explained in section 3, the background model we propose in the DF framework is also built in a data-driven way.

An important shortcoming of both [22] and [11] is that they don't provide smooth degradations to spatial shifts. This is, using these descriptors to compare two identical images that are translated with respect to each other would not provide in general a smooth decreasing distance for their alignment. This obstacle is often overcome in the vision literature by spreading the information spatially blurring the image. There is no straightforward mechanism to achieve this in an integrated way with the probabilistic model.

Blurring or multi-resolution techniques are used in many vision applications, including optical flow [2] and image matching (for example, using the Gaussian pyramid [5]). Blurring provides smoother spatial degradations. This is, these methods spread information in an image so that gradient descent algorithms have a larger "capture range" or basin of attraction, and to avoid local minima of the matching function.

A second purpose of blurring is to mitigate the effects of appearance changes on comparison metrics. If two images cannot be put in perfect pixel-to-pixel correspondence before applying an image difference function like the sum of squared differences, then pre-blurring the images will reduce the effect on differences of these slight misalignments. This type of blurring can be useful in tracking applications when the tracked object is changing appearance from frame to frame.

Unfortunately, blurring images has unwanted side effects. In particular, blurring images creates new image values that are averages of the original values. It is difficult to distinguish, in a blurred image, whether a pixel value of 50 represents several original pixel values of 50 or whether this blurred value is the result of taking the mean of some 0 and 100 pixels. This well-known loss of information from blurring[1] affects the discriminative power of blurring methods, as we show in section 4.

Another use of DFs was demonstrated by Learned-Miller [16] in developing the congealing framework for joint image alignment. In this work, each image is transformed to increase its likelihood with respect to a field of distributions defined across all of the other images. In essence, the likelihood of each image is maximized with respect to the disribution field defined by the set of images at each time step. It has been noted by Learned-Miller [16] that congealing has a large basin of attraction for alignment and that it is relatively robust to local minima in alignment, since a large collection of images can smooth the optimization landscape.

Descriptors for template matching methods can also be interpreted as DFs. The HOG descriptor [10] is a histogram over the gradient of a patch, and therefore a distribution field.

[1] See, for example, the discussion in Szeliski's book [23], page 450.

If computed densely (one histogram at each pixel), since the histogram is computed using a large patch around the pixel, the information about a pixel is spread out, yielding a smoother function. This large ”support” of the descriptor is similar to the blur of the image. This is also in the case of the SIFT descriptor [18]. These descriptors are robust to certain noise, and misalignments. However, the size of the support can't be adapted to the images, or updated for refining the search. These descriptors strongly represent the geometry of the image and can be overly sensitive to changes in pose.

An alternative to building a template is use a histogram that describes the full patch (for example, a face that is being tracked [15]). Histograms provide a larger basin of attraction, and they overcome the problem of oversensitivity to spatial structure. The histogram can be weighted in order to give more relevance to the points in the center of the patch [9, 7], since they are more likely to contain reliable information. These kernel-based methods have had a lot of success because they are fast, simple, and invariant to many pose changes. However, the histogram of a patch is not invariant to changes in illumination. 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 as described in [14] and also prevents some motions like rotation to be modeled. In order to resolve these ambiguities, the size of the descriptor should be expanded. Some recent efforts have focus on including some spatial information in the descriptor by using multiple kernels [14], or multiple patches [12]. These methods improve the performance of the single histogram descriptor, but require other mechanisms to decide the number and shape of the kernels. In [13], the authors propose a solution for the placement of the kernels, but a change in object appearance may make these kernels unstable, and therefore the optimal placement will need to be recomputed. Also, if the number of these kernels is small, the tracking accuracy is more vulnerable to occlusion or abrupt motion. Other additions to the single weighted histogram descriptor are higher order statistics [3] or temporal information [6], and increasing the discriminability of the descriptor using feature selection [8, 17, 25]. DFs can combine the rich and robust information contained in a histogram while preserving the spatial structure of the object to be tracked.

The idea of blurring information only of similar channels is also described in the bilateral filter [24].

3 Description of Distribution Fields

A distribution field (DF) is simply an array of probability distributions, one for each location in a “field”. The probability distribution at each location defines the probability of each feature value at the location. For example, if the feature space were gray-scale intensity, then each location in the field (pixel position) would have a probability distribution over the values 0-255.

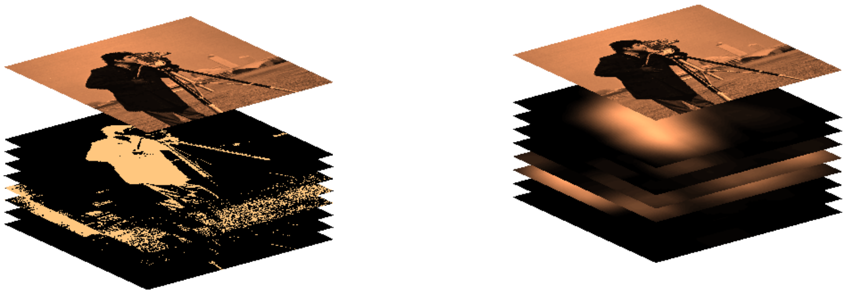

Figure 1: **Left.**. Distribution field after exploding the "cameraman" image. (The original image is shown superimposed on the distribution field for clarity.) The number of brightness levels (or layers) has been quantized to 8. **Right.** The same distribution field after smoothing in the dimensions of the original image.

3.1 Representation of Distribution Fields

A distribution field is represented as a matrix df with $(2+n)$ dimensions, where the first two dimensions are the width and height of the image (or image patch), 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 x n yields a 3D distribution field matrix of size m x n x b, where b is the number of intensity feature values, or bins. For a grayscale image, we can use $b = 256$, or a coarser representation with fewer bins to save memory or processing time. For a higher dimensional feature space, such as two-dimensional gradient measurements, we can build a distribution field with a two-dimensional distribution at each pixel location, yielding a distribution field matrix of four dimensions.

3.2 Exploding an image

Most previous work which used the DF data structure created each distribution field from a set of images. For example, in background subtraction, they are typically created from a video sequence of background images. The DF was then a summary of the statistics of the set of background images. In this work, we use DFs as descriptors for single images. The first step in creating a DF from a single is image is what we call *exploding an image* into a distribution field.

Exploding an image into a distribution field results in a distribution field with a Kronecker delta function at each pixel location. In particular, exploding an image I into a distribution field df with as many bins as features values is defined by

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

where i and j index the row and column of the image, and k indexes the possible values of the pixel. We call each of the k bins 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 1 shows the results of computing this DF for the well-known camera man image.

At this point, the distribution field representation contains exactly the same information as the original representation, albeit in a much larger representation. We now show how to "spread" the information in the image without destroying it as occurs in traditional blurring techniques.

3.3 Smoothing Distribution Fields

The blurring of distribution fields has the same benefits as blurring images, but few of the drawbacks. The right side of Figure 1 shows a smoothed version of the distribution field on the left. The three dimensional distribution field matrix has simply been convolved with a *two-dimensional* Gaussian filter which spreads out in the x and y dimensions, but not in the feature dimension. That is, the smoothed distribution field df_s can be written simply as

$$df_s = df * h_{(x,y)}, \tag{1}$$

where $h_{(x,y)}$ is a 2D Gaussian kernel, and $*$ is the convolution operator.

Prior to convolution, we could interpret any value of 1 in layer L of a distribution field to mean "there is a pixel of value L at this location in the original image." After convolution, we have a somewhat different interpretation. The semantics of the smoothed distribution field 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 averaging 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 distribution field df_s maintains the property that each column of pixels integrates to 1,[2] and hence is still properly called a DF.

Smoothing can also take place in feature space. This allows the model to explain changes due to subpixel motion, shadows, and changes in brightness. In the example of a grayscale image, this smoothing is a 1D Gaussian filter over the third dimension. We can write a further smoothing df_{ss} of the already spatially smoothed distribution field df_s as

$$df_{ss} = df_s * h_z, \tag{2}$$

where h_z is a 1D Gaussian kernel along the feature dimension.

3.4 Comparison between Distribution Fields

The comparison between DF's that different images yield can be done with any distance function like L1 and L2 over the difference between the two matrices df_1 and df_2. For

[2] This property breaks down at the boundaries of images.

example, an L_1 difference would be defined as

$$D_{L1}(df_1, df_2) = \sum_{i,j,k} |df_1(i,j,k) - df_2 i, j, k|. \tag{3}$$

Our experiments on basins of attraction in section 4 show that the performance does not strongly depend upon the difference function used.

Also, using their probabilistic interpretation, the comparison between DFs can be done by computing the likelihood of one being sampled from the other. Given the two images I_1 and I_2, the likelihood that I_2 came from I_1 is

$$P(I_2|I_1) = \prod_{x \in I_2} P(x|I_1), \tag{4}$$

which in the DF framework can be computed as

$$P(I_2|I_1) = \prod_{x \in I_2} (DF_1(x) * G(\sigma))DF_2(x), \tag{5}$$

where both DF_1 and DF_2 are binary DFs and $G(\sigma)$ is a 3-dimensional Gaussian filter with standard deviation vector σ.

3.5 Sharpening match

When two images are compared using the likelihood function described in (5), a decision needs to be made about the value of the parameter σ. If σ is very large, all DFs will be very similar, because the distributions will be close to uniform. If σ is too small, small spatial shifts or changes in illumination might produce a much worse similarity value between the images than it should. This value of σ needs to be adaptative to each pair of DFs. The best value of σ for comparing two images can be computed by maximizing its likelihood

$$L(\sigma|I_1, I_2) = \prod_{x \in I_2} (DF_1(x) * G(\sigma))DF_2(x) \tag{6}$$

which can be expanded to

$$L(\sigma|I_1, I_2) = \prod_{i \in I_2} \sum_{j \in I_1} \frac{1}{(2\pi)^{(3/2)}} \frac{1}{\sigma_s^2 \sigma_f} e^{-\frac{1}{2}\left(\frac{(x_j - \mu_{x_i})^2 + (y_j - \mu_{y_i})^2}{\sigma_s^2} + \frac{(z_j - \mu_{z_i})^2}{{\sigma_f}^2}\right)}, \tag{7}$$

where σ_f and σ_s are the standard deviations of the Gaussians filter used to convolve in feature space and image space respectively. Changing the product for the sum of the logs and rearranging the terms, the expression for the log-likelihood is

$$\log L(\sigma_s|I, t) = |t| \log\left(\frac{1}{(2\pi)^{(3/2)} \sigma_s^2 \sigma_f}\right) + \sum_{i \in t} \log\left(\sum_{j \in I} e^{-\frac{1}{2}\left(\frac{(x_j - \mu_{x_i})^2 + (y_j - \mu_{y_i})^2}{\sigma_s^2} + \frac{(z_j - \mu_{z_i})^2}{{\sigma_f}^2}\right)}\right). \tag{8}$$

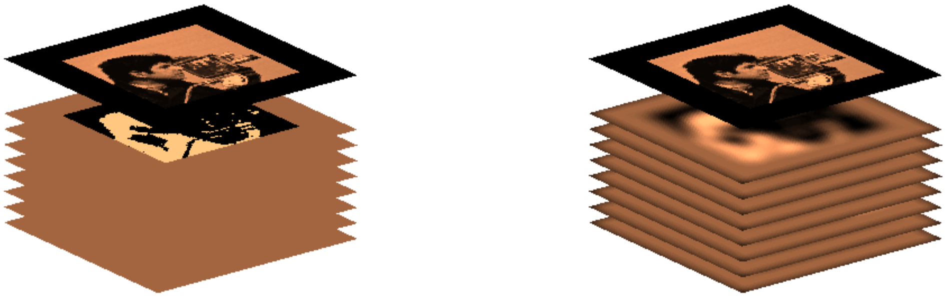

Figure 2: This shows how a patch is surrounded by a set of uniform distributions to handle the boundary case in smoothing patches.

3.6 Blending Distribution Fields

Because we build smooth distribution fields by convolving with Gaussian kernels, we have to deal with boundary issues at the edges of patches or images. To do this, we assume that we have no knowledge of the image outside of a patch. But we would like to spread the information from a patch beyond the original patches borders. To achieve this, we surround an exploded distribution field with a "uniform" distribution field that consists of uniform probability distributions. Then, the process of convolutional smoothing described earlier has the effect of mixing the uniform distributions outside the patch with the brightness distributions within the patch to create a smooth transition across the patch boundary. This process is illustrated in Figure 2.

3.7 Combination of Distribution Fields

In some cases, combining the information of several DF's can be useful. For example, in combining information about the past in tracking, we use a linear combination of an old distribution field and a new distribution field. This operation in DF's can be implemented as the component-by-component average of the matrices. This allows us to build a non-parametric model based on the real distribution at each pixel. This can also be useful for tracking or joint alignment.

4 Basin of attraction

One of the most important advantages of DFs is the capacity to spread out information about a pixel value in image space without loss of this information. This slow degradation to spatial location is useful for finding correspondences because it creates a wide basin in the objective function around the local optimum. This is very good in cases of large displacement or sudden move. In order to measure the width of this basin of attraction, we have run several experiments.

4.1 Why the basin of attraction is important for tracking

Naively, finding correspondences in an image could be done using a sliding window strategy (exhaustive search) for each image patch. However, search for correspondences in tracking, flow, alignment, etc can be efficiently done through gradient descent, by taking advantage of the smooth motion assumption. This search strategy relies on the assumption that there isn't any local optimum in the path traveled by following gradient descent from the start of the search to the global optimum. This might not be the case if the start of the search (or first guess) is too far from the global optimum, or if the objective function is not smooth. In this section we focus on analyzing this last condition of smoothness. In order to evaluate the smoothness of the objective function around the global optimum we use the concept of basin of attraction.

Given an objective function $f(x)$ and a point p, we define the *basin of attraction* as the neighborhood of positions around p from which following the gradient of $f(x)$ leads to p. The width of this basin of attraction around the true position of a target sits at the core of the problem of finding correspondences, and serves evaluating the goodness of the objective function for gradient descent search methods. Objective functions with wider basins of attraction will be more robust to poor initial guesses and will be more likely to converge to the true position of the target.

In this section we describe a series of experiments that evaluate the width of the basin of attraction of different descriptors, features spaces and similarity metrics.

4.2 Experiment details

This semi-synthetic experiment isolates the problem of creating a smooth objective function to that of creating an objective function robust to changes in appearance.

The experiment takes an image and an arbitrary patch in it (of size $30x30$), and evaluates the basin of attraction around the true position of this patch, for a given descriptor, a given feature space and a given similarity metric. Notice that if the search strategy in this experiment was exhaustive search, the experiment would be trivial.

The general method to compute the basin of attraction is as follows. We arbitrarily select the patch whose top left corner is in position $p = (x, y)$. The search starts by comparing it with the patch whose top left corner is in position $p = (x, y - 1)$. If the search converges to the true position, then we try starting in $p = (x, y - 2)$, then $p = (x, y - 3)$, etc, up to $p = (x, y - max_d)$. In our experiments, we have chosen $max_d = 30$. The first displacement d_r for which the search doesn't converge to the true position marks the width of the basin of attraction. This is also done in the other horizontal direction, by starting the search in $p = (x, y+1)$, then $p = (x, y+2)$, etc., to obtain the width of the basin in the other direction d_l. The width of the basin is the minimum of d_r and d_l.

The details of the different combinations of search strategies, similarity metrics, parameter settings and descriptors are described below.

- **A: L1 distance** The descriptors used to represent the image and the patch are DF's over intensity space, with $nbins = 256$. Both DF's are convolved with a Gaussian filter with parameter $\sigma = 15$. The similarity metric for comparing two DF's is L_1. The search strategy used is gradient descent. For a particular displacement d, the steps to decide whether $p - d$ is within the basin of attraction are:

  1. Explode and smooth full image.
  2. Explode and smooth patch.
  3. Use L_1 to follow gradient descent until minimum is reached. For two distribution fields df_1 and df_2, the distance metric L_1 is defined as in equation (9).

  $$L_1(df_1, df_2) = \sum_{i,j,k} |df_1(i,j,k) - df_2(i,j,k)| \tag{9}$$

- **B: Maximum Likelihood with fixed** σ. The previous approach requires the value of σ to be fixed. This can be avoided by choosing σ such that it maximizes the likelihood of a DF with respect to the other. This likelihood is the product of the probabilities of each point in the first DF under the probability distributions in the second DF. This is described in detail in equation (7). This experiment tests the basin of attraction of DF's when σ is chosen automatically. The descriptors used are also DF's over intensity space, with $nbins = 256$. Only the DF that represents the full image is convolved. The value of the parameter σ used is the one that maximizes the likelihood. The similarity metric for comparing two DF's is the likelihood of one with respect to the other, from equation (7). The search strategy is gradient descent. For a particular displacement d, the steps to decide whether $p - d$ lies within the basin of attraction are:

  1. Explode full image.
  2. Explode patch.
  3. Compute the σ_{max} that maximize the likelihood of the patch with respect to the full image, according to equation (7). This computation is exhaustive over a discrete, finite set of different values. In this experiment, these are $1, 3, 5, ..., 39$.
  4. Smooth full image with σ_{max}.
  5. Use the likelihood of the DF's to follow gradient descent, until a maximum is reached.

- **C: Maximum Likelihood with single-pixel steps**. The value of the optimal sigma varies depending on the displacement from the true position. In general, the smaller the displacement, the more similar their DF's are and the smaller sigma needs to be in order to maximize the likelihood. In this case, the value of sigma should be updated at each step in the gradient ascent, and this is what this experiment tests. In this experiment everything is the same as in (B), except that the value of σ is updated at each step. For a particular displacement d, the steps to decide whether $p - d$ is within the basin of attraction are:

1. Explode full image.
2. Explode patch.
3. Compute the σ_{max} that maximize the likelihood as in (B).
4. Smooth full image with σ_{max}.
5. Use the likelihood of the DF's to follow gradient ascent for 1 step.
6. Repeat 3-5 until a maximum is reached.

- **D: Maximum Likelihood with single-pixel steps. σ chosen using gradient ascent**. The optimization of σ can be time consuming if done exhaustively. However, the likelihood function is smooth enough so that following gradient ascent given a good start can lead to a good value of σ. This experiment is identical to (C), but instead of exploring the full range of likelihoods for the values $1, 3, 5, ..., 39$, σ_{max} is chosen using gradient ascent.

- **E: Maximum Likelihood with multi-pixel steps**. Optimizing the value of σ at each step is computationally intensive. Instead, we can alternate between the optimization of position and σ. This experiment is identical to C, except that the optimization of position and σ alternate. For a particular displacement d, the steps to decide whether $p - d$ is within the basin of attraction are:

  1. Explode full image.
  2. Explode patch.
  3. Compute the σ_{max} that maximizes the likelihood as in (B).
  4. Smooth full image with σ_{max}.
  5. Use the likelihood of the DF's to follow gradient ascent until a maximum is reached.
  6. Repeat 3-5 until a maximum is reached.

- **F: Other descriptors**. For the sake of comparison we also compute the basin of attraction of commonly used image descriptors. These are normalized cross correlation, blurring the image, mean-shift and multiple kernel tracking.

  1. **Normalized Cross Correlation.** The procedure to calculate the basin of attraction is the one described before of using increasingly larger displacements. For a particular displacement d, the point $p - d$ lies in the basin of attraction if the

gradient descent starting in $p - d$ converges to p. The objective function of the gradient descent for two image patches I_1 and I_2 is

$$NCC(I_1, I_2) = \frac{\sum_{x,y} \left(I_1(x,y) - \bar{I}_1\right)\left(I_2(x,y) - \bar{I}_2\right)}{\sqrt{\sum_{x,y} \left(I_1(x,y) - \bar{I}_1\right)^2 \sum_{x,y} \left(I_2(x,y) - \bar{I}_2\right)^2}}, \tag{10}$$

where $\bar{I}$ is the mean of the pixel values of image I.

2. **SSD using image blur.** The procedure to calculate the basin of attraction using blur is the same as the one for NCC, except that the objective function is

$$SSD_{blurred} = \sqrt{\sum_{x,y} I_1(x,y)' - I_2(x,y)'}, \tag{11}$$

where I' is the result of blurring the image by convolving it with a Gaussian filter ($I' = I * G(\sigma)$). In our experiments, we use $\sigma = 15$, which is the same size as in the DF experiment.

3. **Mean Shift.** Mean Shift tracking is very similar in spirit to DF since it uses a histogram-based descriptor. Instead of using the mean shift search strategy, we use gradient descent. The method to compute the basin of attraction is identical to the SSD and NCC, except that the objective function is the Bhattacharyya coefficient of the weighted histograms of the image patches. The weight of the histograms is computed as described in [9].

4. **Multiple Kernel Tracking.** This method is similar to mean shift, except that there multiple weighted histograms per image patch. Each of these weighted histograms is generated using a different kernel to compute the weights. We chose a roof kernel and a gaussian kernel, as described in the original paper [14].

The data used in these experiments comes from the UWA database available in `http://www.cs.washington.edu/research/imagedatabase/groundtruth/`. A subset of 289 images were arbitrarily selected for these experiments.

4.3 Experiments discussion

Figure 3 shows the results of the basin of attraction for several similarity metrics compared to other traditional ones. For each number in the x axis, It shows the ratio of images that have a basin of attraction of at least that number. It's surprising to observe the performance of the method that uses a fixed sigma obtained by maximizing the likelihood. In fact, the graph can be misleading, since what it's showing is not a much narrower basin of attraction around the maximum, but lack of accuracy. On average, the position of maximum likelihood using this technique is around 6 pixels away from the true position. Figure 4 shows some examples of the performance of this technique. Figure 5 shows the the results for the rotation experiment.

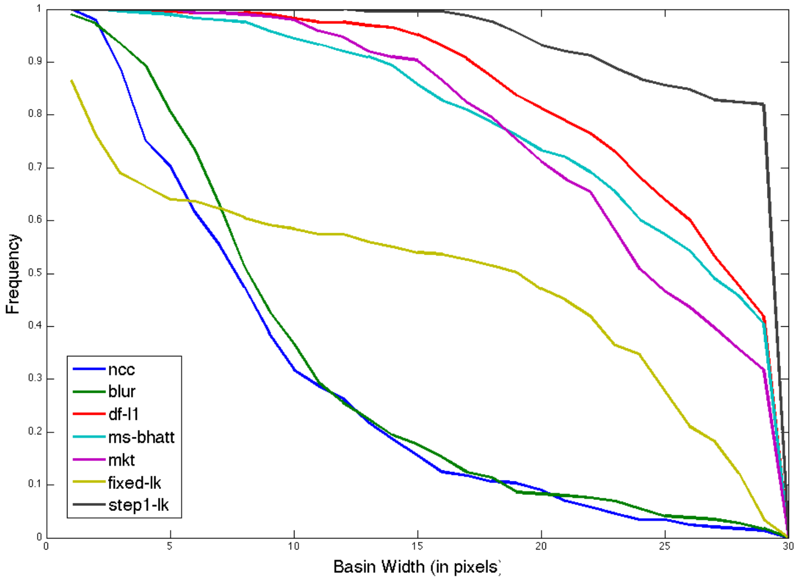

Figure 3: Basin of attraction of different similarity metrics

5 Tracking

5.1 Tracking Algorithm using Distribution Fields

Since the DF descriptor provides a large basin of attraction, the algorithm used for tracking is fairly simple. There are a few parameters in our model, all of which were set by experimenting on a set of training videos that were independent of the test set.

The model of the target is created by exploding the image that contains the target into a distribution field and smoothing the distribution field with a Gaussian kernel. Searching for the target in a new frame consists of building a new DF by also exploding and creating a DF at each candidate position, and following the direction where the gradient of the L1 difference between the two DF's 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:

$$df_{model}^{(t+1)} = \lambda df_{model}^{t} + (1-\lambda) df_{obs}^{(t+1)}. \tag{12}$$

In our work we use $\lambda = 0.95$, based on experiments with training videos.

In order to make the algorithm more robust to the values of σ used to smooth the DF's, we use a hierarchical approach. Instead of using a single DF to represent the target, we use a small set of DF's, where each of them is built using an increasing value of σ. These DF's contain information at different frequencies, resembling a Gaussian pyramid. At each frame, we use a coarse to fine strategy. The most smoothed DF is used to start search, until it reaches a local minimum. This position is the start for the search using the second DF.

We summarize the procedure in pseudo-code:

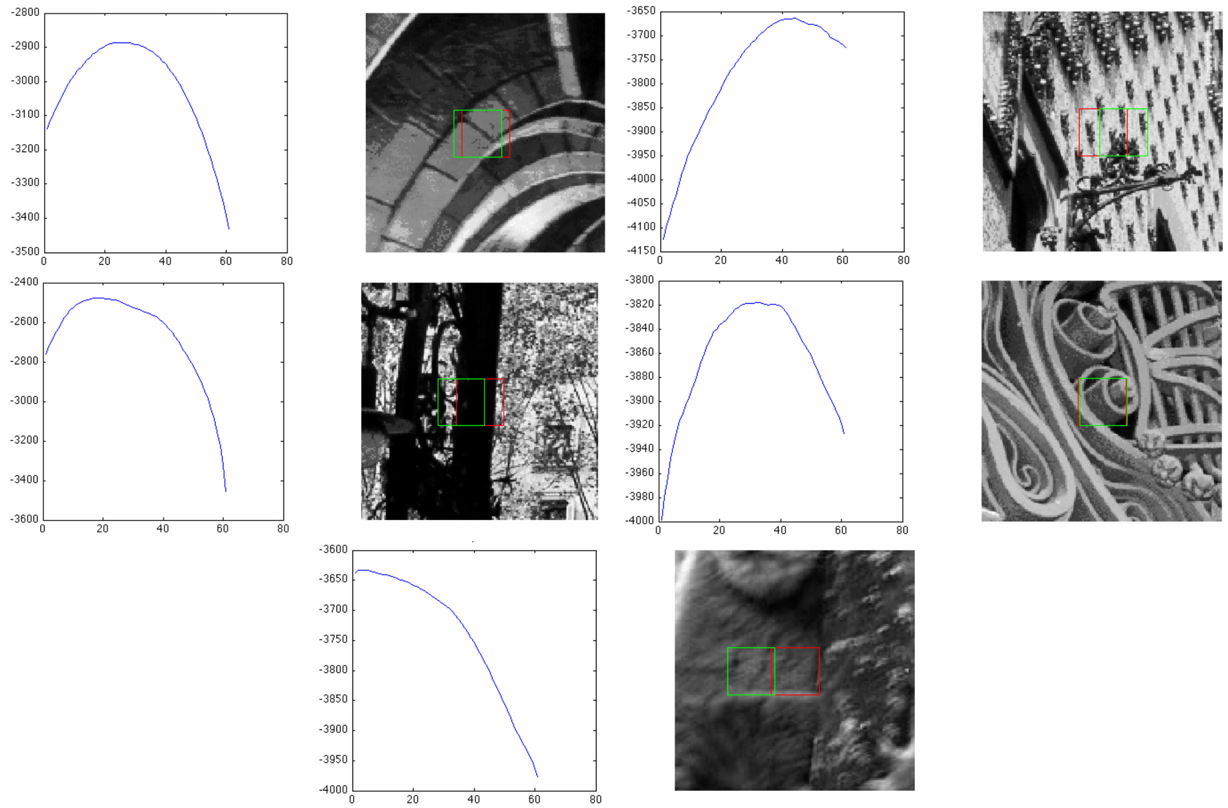

Figure 4: Basins of attraction and local optima reached for different sample images. Each graph shows the likelihood along the x axis, the true position is at 31. Each image shows the true position of the patch in red and the local optimum reached in green. It seems reasonable that patches with rich texture (like the one in second row and second column) will be easier to find than those more homogeneous (like that one in the third row). Repeating patterns are likely to be confusing (as in the first row, second column), and some mistakes are surprising (like the one in second row first column). A typical example is the very first one, where the true position is missed by 5-6 pixels. One possible explanation is that the highest frequency information is lost when the DF is convolved with a very large value of σ.

```
1. Input: Video sequence $V$ and patch $I$ in frame 1.
2. Parameters, all set using training video sequences.
    (a) Spatial smoothing parameters $\sigma_{xy}^{i}, i \in 1, 2, 3$.
    (b) Brightness smoothing parameter $\sigma_z$.
    (c) Number of brightness bins $b$.
    (d) Mixing parameter $lambda = 0.95$.
3. Initialize $df_{model}^{0} = explode(I) * h_2 * h_1$
4. Initialize target location $(x, y)$ to center of patch $I$.
5. For each video frame:
    (a) $df_{object} = explode(patch(x, y)) * h_2 * h_1$.
```

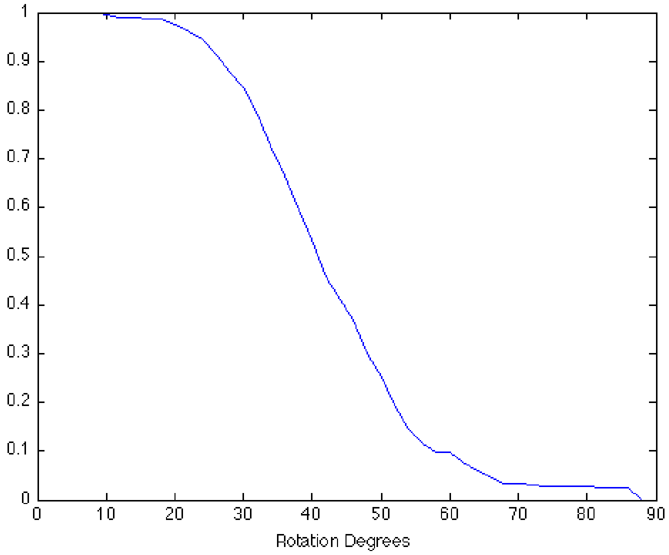

Figure 5: Results for the basin of attraction in rotation.

(b) Minimize difference $D(df_{object}, df_{model})$ using gradient descent of patch position. Recalculate df_{object} for each new location.

(c) Reset (x, y) to result of gradient descent.

(d) $df^{t+1}_{model} = \lambda df^{t}_{model} + (1 - \lambda) df_{object}$.

5.2 Tracking Experiments

Two key questions must be answered to perform evaluations: which methods to compare against and which test suite to use. Unfortunately there does not seem to be a widely agreed upon standard for a tracking test suite in the computer vision literature. Many algorithms are shown running on videos that are not available publicly or have not been used by other researchers.

Recently, however, Babenko et al. [1] compiled a list of commonly used and publicly available videos[3] that have been used to assess tracking algorithms. While these videos do not represent the full gamut of scenarios under which to evaluate tracking, they nevertheless 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 backgrounds, and complex backgrounds (surfing). They also show a variety of objects being tracked such as faces, toys, and soda cans.

Two of the recently top performing tracking algorithms, the MIL tracker [1] and PROST [19] have been run on 6 of these videos, and MIL has been run on all of them. For this reason we chose to evaluate our algorithms on these videos. We shall refer to this collection of 11

[3] The videos can be found at `http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml`.

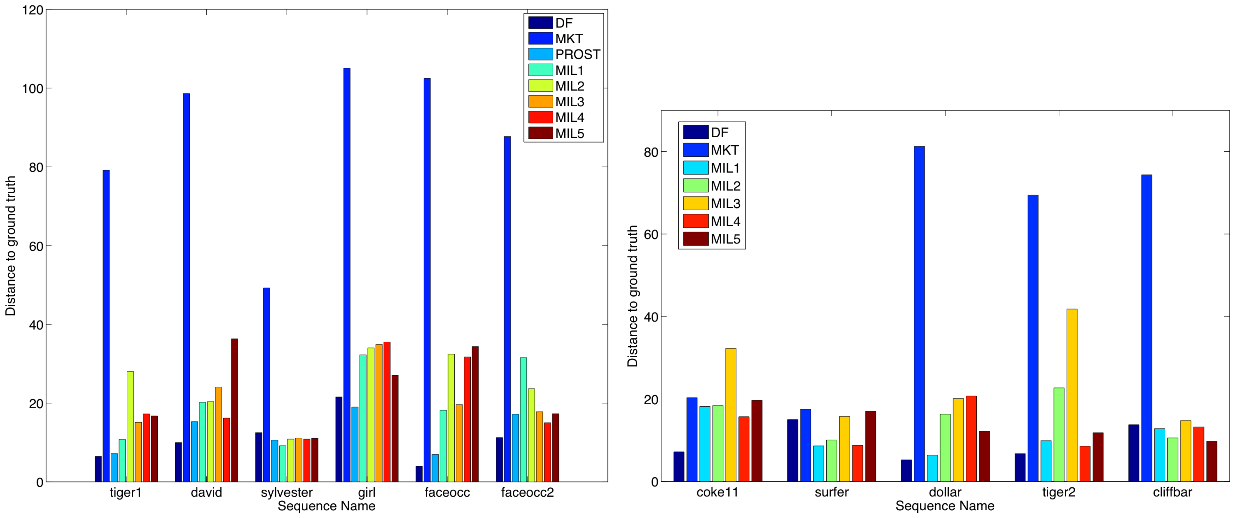

Figure 6: **Mean distance to target (shorter is better)**. Performance comparison between DF tracking and other algorithms. Both graphs show the average distance between the estimated object location and the location marked as ground truth. The plots on the left show results for videos in which PROST results were available. The plots on the right show results for which PROST results were not available. We outperform PROST in 4 of the 6 videos. We far outperform all other algorithms. Since the MIL tracker is non-deterministic, we show results for 5 separate runs.

videos[4] as the MIL video database.

Among the algorithms that have been evaluated on the MIL video database, the best performers have been the MIL tracker [1] and the PROST tracker [19]. For this reason, we included these two trackers in our comparisons. Unfortunately, PROST has only been run on a subset of six of the videos, and therefore we can only compare directly to PROST on half of the videos.

The MIL tracker uses unsupervised online learning over Haar features to train a discriminative model over the two classes background and foreground. They improve over other ”learning-by-detection” algorithms by not requiring the labeling of data, which may be inaccurate by using boosting instead. In [19], three different trackers are used: one uses normalized cross correlation directly over the intensity values, the second one uses optical flow and mean shift, and the last one is an online random forest. The three trackers are combined using a simple cascade. In this paper we use PROST and MIL as representatives to compare in our experiments because they are, to the best of our knowledge, the algorithms that have the highest performance in this dataset.

In addition to these trackers, we felt that our tracker was most similar in spirit to the Multiple Kernel Tracker [14], and so we implemented this tracker and also ran it on the same suite. The kernels used in our implementation are those described in their experiments, which

[4] There is a 12th video, but the link to it is broken.

are an Epanechnikov kernel and a roof kernel.

We evaluate our algorithm and other usings two different modes of comparison. The first, for which results are shown in Figure 6, is the mean distance, over all frames in the video, of the distance between the algorithm's estimated target location and the ground truth location. While this is certainly a useful metric, it has some negative elements. For example, if a tracker "falls off" the target and essentially has no idea where it is, then its subsequent score will simply be determined by how far it happens to be from the tracked object. If it is "unlucky" and happens to be on the other side of the image, it will suffer much more than if it happens to be near the target, even though it has no knowledge of the target in either case.

The plots on the left side of Figure 6 include comparisons with PROST, the MIL tracker, and the MKT tracker. For the PROST tracker, we show improved performance (lower mean distance) on 4 of the 6 videos, although admittedly these gains are small. In addition to outperforming the PROST tracker, we believe our algorithm is significantly simpler, since it relies on only a single descriptor (rather than three) and there is no arbitration among trackers, as int he PROST tracker.

For the MIL and MKT trackers, we show significantly improved results. Because the MIL tracker is stochastic, we show 5 runs of it. The right half of the figure shows additional results for the MIL and MKT trackers. PROST results were not available for these videos.

References

[1] B. Babenko, Ming-Hsuan Yang, and S. Belongie. Visual tracking with online multiple instance learning. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 0:983–990, 2009.

[2] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56:221–255, 2004.

[3] Stanley T. Birchfield and Sriram Rangarajan. Spatiograms versus histograms for region-based tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1158–1163, 2005.

[4] Thomas Brox and Jitendra Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:500–513, 2011.

[5] Peter J. Burt, Edward, and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[6] Kevin J. Cannons, Jacob M. Gryn, and Richard P. Wildes. Visual tracking using a pixelwise spatiotemporal oriented energy representation. In *Proceedings of European Conference on Computer Vision: Part IV*, pages 511–524, 2010.

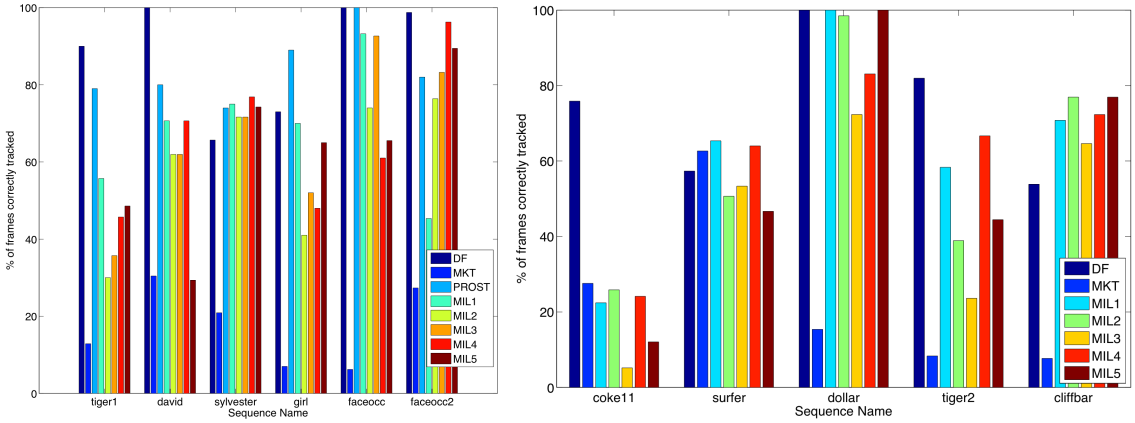

Figure 7: **Percent of frames correctly tracked (higher is better)**. Performance comparison between distribution field tracking and other algorithms. Both graphs show the percentage of frames "correctly tracked." For a particular frame, with ground truth defined by rectangle A and estimated object position defined by rectangle B, the object was judged to be correctly tracked if the ratio of the areas of the intersection of A and B to the union of A and B was greater than 0.5. The graphs on the left are data for which the PROST tracker results were available. We outperform the PROST tracker in 3 videos, tie in 1, and lose in 2. We easily outperform the other trackers on average. IThe plots on the right are data for which the PROST tracker results were not available. As in Figure 6, we show five separate runs of the MIL tracker.

[7] Robert T. Collins. Mean-shift blob tracking through scale space. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2:234, 2003.

[8] Robert T. Collins and Yanxi Liu. On-line selection of discriminative tracking features. *Proceedings of the International Conference on Computer Vision*, 1:346, 2003.

[9] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2:2142, 2000.

[10] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1:886–893, 2005.

[11] Ahmed M. Elgammal, David Harwood, and Larry S. Davis. Non-parametric model for background subtraction. In *Proceedings of the European Conference on Computer Vision-Part II*, pages 751–767, 2000.

[12] Zhimin Fan, Ming Yang, and Ying Wu. Multiple collaborative kernel tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[13] Zhimin Fan, Ming Yang, Ying Wu, Gang Hua, and Ting Yu. Efficient optimal kernel placement for reliable visual tracking. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, pages 658–665, 2006.

[14] Gregory Hager, , Gregory D. Hager, Maneesh Dewan, and Charles V. Stewart. Multiple kernel tracking with SSD. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 790–797, 2004.

[15] Thomas Riisgaard Hansen, Eva Eriksson, and Andreas Lykke-Olesen. Use your head: exploring face tracking for mobile interaction. In *CHI '06 extended abstracts on Human factors in computing systems*, CHI EA '06, pages 845–850, New York, NY, USA, 2006. ACM.

[16] Erik G. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:2006, 2006.

[17] Alex Po Leung and Shaogang Gong. Mean shift tracking with random sampling. In *Proc. BMVC 2005*, pages 729–738, 2006.

[18] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, November 2004.

[19] Jakob Santner, Christian Leistner, Amir Saffari, Thomas Pock, and Horst Bischof. PROST: Parallel robust online simple tracking. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 0:723–730, 2010.

[20] H. Sekkati and A. Mitiche. Joint optical flow estimation, segmentation, and 3d interpretation with level sets. *Comput. Vis. Image Underst.*, 103:89–100, August 2006.

[21] Yaser Sheikh and Mubarak Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1778–1792, 2005.

[22] Chris Stauffer and W. Eric. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

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

[24] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the International Conference on Computer Vision*, pages 839–, 1998.

[25] Zhaozheng Yin, Fatih Porikli, and Robert T. Collins. Likelihood map fusion for visual object tracking. In *Proceedings of the 2008 IEEE Workshop on Applications of Computer Vision*, pages 1–7, 2008.