Learning and Matching Multiscale Template Descriptors for Real-Time Detection, Localization and Tracking

Taehee Lee Stefano Soatto
Computer Science Department
University of California, Los Angeles, CA 90095
{taehee,soatto}@cs.ucla.edu

Abstract

We describe a system to learn an object template from a video stream, and localize and track the corresponding object in live video. The template is decomposed into a number of local descriptors, thus enabling detection and tracking in spite of partial occlusion. Each local descriptor aggregates contrast invariant statistics (normalized intensity and gradient orientation) across scales, in a way that enables matching under significant scale variations. Low-level tracking during the training video sequence enables capturing object-specific variability due to the shape of the object, which is encapsulated in the descriptor. Salient locations on both the template and the target image are used as hypotheses to expedite matching.

1. Introduction

We tackle the problem of learning, detecting, localizing and tracking individual objects for use in augmented reality, navigation, and general real-time scene analysis from a moving hand-held platform, such as a phone.

Rather than addressing the general visual recognition problem, the application context drives our approach. In particular, we do not address intra-individual variability, and focus on learning – and later detecting and localizing – *individual objects*, rather than object categories as in challenges such as PASCAL [3]. This does not make the problem trivial, as the data still exhibit significant nuisance variability due to large changes in vantage point, illumination, and occlusions between the training and the test sets. We focus, therefore, on handling such a variability in a computationally efficient manner (Section 2), while making informed choices on the compromises implicit in our modeling assumptions (Section 3).

Unlike standard object categorization using web-collected datasets, in our context we typically have *video sequences* as training sets, captured purposefully by a human, *e.g*. by waving the phone in front of an object of interest. Since the users' goal is to "explore" the object for later recognition, such a purpose is usually reflected in the video containing a fair sample of the nuisance distribution, as we will see in the discussion leading to eq. (10). Although our approach still entails a loss compared to marginalization proper, at least we take into account the class-specific distribution of the nuisance variables in learning the template, unlike generic local features computed individually in single images.

1.1. Related Work

Our work relates to much of the recent literature on fast sliding-window classifiers, and tracking-by-detection, as we discuss throughout the paper. However, we take full advantage of the available video data during training, and avoid exhaustive window search.

Recent contributions to the literature include [7], that generates an efficient implementation of mean-patch descriptors, and [8] that focuses on detecting textureless objects using dominant orientations, and utilizing bit-wise operations with SIMD instructions for fast histogram matching, like [20]. The latter selects feature points from warped images and uses them to compute local descriptors. Learning is time consuming which makes this approach not well suited to our applications. [12] learns randomized trees for recognizing keypoints for 3D object detection. Since we consider scalability in memory for mobile platforms, here we focus on templates as our descriptors. In order to handle variations of the template while tracking, [9] showed an efficient and robust online learning mechanism for detection and tracking. They used positive and negative samples to make their descriptors more robust and generate multiple templates for different appearances of the target while tracking. They use a sliding window search algorithm to localize the object. Online learning and boosting approaches are also shown in [5, 1]. In [5], they proposed an on-line AdaBoost for feature selection applied to object detection and tracking. In [1], another robust tracking mechanism is introduced. They take multiple patches around a target object and train instance classifiers using online boosting. However, they do not consider scale changes in their motion model, and their tracking diverges when the target moves with significant scale changes.

To reduce the number of test windows for localization, [10] introduced an efficient branch-and-bound approach for subwindow search that guarantees a global maximum, with the help of a quality bounding function. They show their algorithm applied to an object localization system that computes local descriptors such as SIFT [13] and quantizes the descriptors to build classifiers with spatial pyramid kernels. We share the goals to introduce efficient features and object tracking for mobile platforms with [21, 6]. In [21], they utilize optimized features to achieve real-time performance on mobile phones. [6] introduces a contour-based shape descriptor for recognition and pose estimation.

In this paper, we focus on efficiently matching template descriptors under significant scale changes while exploiting training video. We learn a descriptor that captures image statistics from multiple video frames by taking advantage of low-level tracking and sampling from multi-scale regions. Finally, detecting an object at multiple scales is performed efficiently by using salient locations in the descriptor and the test image, instead of searching over the whole image domain by sliding a window.

2. Method

We consider image sequences at multiple scales, indicated by $I_t(x, y; \sigma)$, whereby $I : \mathbb{Z}^+ \times \mathbb{Z}^{N\times M} \times \mathbb{Z}^+ \longrightarrow \mathbb{Z}^+$; $(t, x, y, \sigma) \mapsto I_t(x, y; \sigma)$ maps positive times, scales, and points on the $N \times M$ lattice to positive (grayscale) values. When considering the image at the native scale we write $I_t(x, y)$. Within each sequence, for each time t, scale σ and position (x, y), we consider a *window*

$$w_t(x, y; \sigma) \doteq I_t\left(\frac{\tilde{x} - x}{\sigma}, \frac{\tilde{y} - y}{\sigma}\right), \ \forall\, (\tilde{x}, \tilde{y}) \in \Omega(t). \quad (1)$$

While in general the size and aspect ratio of the window can change over time, for simplicity we consider $\Omega(t) = \mathbb{Z}^{88\times 88}$ as our initial window. The size of the image domain covered by the window can change over time due to changes in scale σ, and when the window intersects the boundary of the image domain. Within the window, we consider a sub-sampled lattice, whose nodes we call *centers*, $c_i = (x_i, y_i) \in \Omega \cap \mathbb{Z}^{N_c \times M_c}$. Again, for simplicity, we consider a fixed number of centers, $N_c = M_c = 11$, so c_i, $i = 1, \ldots, 121$. Around each center c_i we consider a *"ball"* (neighborhood) of radius r, $\mathcal{B}_r(c_i) \subset \Omega$ so that the intersection of all neighborhoods covers (with overlap) the entire domain of the window, Ω. Since we have initialized $\Omega = \mathbb{Z}^{88\times 88}$, with 11×11 *centers*, $\mathcal{B}_r(c_i) = \mathbb{Z}^{8\times 8}$ for all $i = 1, \ldots, 121$. Note that as scale σ changes, the size of $\mathcal{B}_r(c_i)$ also changes accordingly.

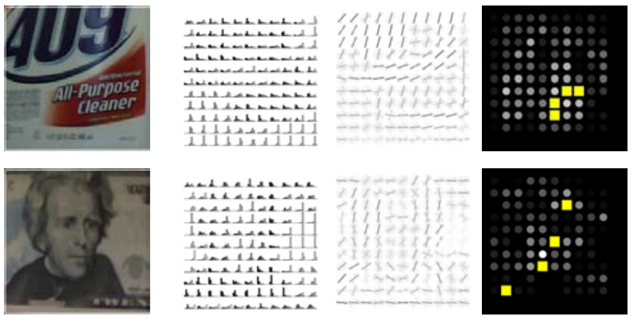

Figure 1. Examples of template descriptor: image, intensity template, gradient orientation template, and feature distribution with anchor points (yellow).

2.1. Training a template

Each window supports a *descriptor*, $\phi_t(x, y)$, that aggregates image statistics within each neighborhood, around each center in the domain of the window, and across all scales. We consider two types of statistics: gradient orientation, and normalized intensity.[1] We aggregate both using a quantized histogram with $K = 8$ bins:

$$h_t^g(i, j) = \{\#(\tilde{x}, \tilde{y}) \in \mathcal{B}_r(c_i) \mid \langle v_j, \frac{\nabla_\epsilon I_t(\tilde{x}, \tilde{y}; \sigma)}{\|\nabla_\epsilon I\|}\rangle \leq \frac{2\pi}{K}\} \quad (2)$$

where v_j, $j = 1, \ldots, K$ are uniformly spaced in the unit circle and ϵ is the size of the kernel used to perform differentiation, which we choose to be $\epsilon = 1$ pixel. We use one of K bins as a special bin for gradients with magnitudes smaller than a threshold, similar to [8], then $v_j, j = 2, \ldots, K$ are uniformly spaced in the unit circle. Since all neighborhoods have the same size, we do not normalize the histogram. Similarly, we have

$$h_t^I(i, j) = \{\#(\tilde{x}, \tilde{y}) \in \mathcal{B}_r(c_i) \mid I_t(\tilde{x}, \tilde{y}; \sigma) = I_j\} \quad (3)$$

where I_j, $j = 1, \ldots K$ are uniformly or log-spaced in $[0, 255]$. The intensity values are normalized by the mean and the standard deviation in the window Ω. Note that the histograms are aggregated across all scales σ, so the histogram can be thought of as an approximation of a mixture density with a mode for each scale. If we collect all K bins for two statistics and all centers c_i, for a window at position (x, y), we can represent the histograms as a 121×16 matrix:

$$\phi_t(x, y) = \{h_t^I, h_t^g\}. \quad (4)$$

[1]Even though the former is a function of the latter, it is a non-linear function computed on an ϵ-neighborhood of each point (due to the regularization implicit in the gradient operator for the discrete lattice), so we expect it to capture some more of the local structure of the image beyond its raw value.

Optionally, a coefficient can be introduced to weight the two histograms differently. Next, we describe how a (static) descriptor (template) $\phi \in \mathbb{Z}^{121 \times 16}$ can be learned from an image sequence $\{I_t\}_{t=1}^T$, and then how a test image I_τ can be compared with the template to detect a previously seen object.

During training, we assume we have a *tracking* procedure in place, whereby the center of a window has been determined: $\{(x(t), y(t))\}_{t=1}^T$. Training is then performed in a straightforward manner by averaging instantaneous descriptors *in the moving frame*, into a static *template*:

$$\phi \doteq \frac{1}{T} \sum_{t=1}^{T} \phi_t(x - x(t), y - y(t)). \tag{5}$$

We use an efficient multi-scale tracker called *TST* [11]. In order to track the window $\{(x(t), y(t))\}_{t=1}^T$ in scale-space, individual point-features are selected in the region-of-interest determined by the user, and tracked across multiple scales. Each track, of variable duration and scale, is used to support a simple translational, similarity or affine transformation of the region $w(x(t), y(t))$ using standard acceptance sampling [4]. Note that, despite organizing the domain of the window w into neighborhoods around each center, data from the window is sampled *densely* during training, in the sense that each pixel in the domain of the window contributes to the descriptor, indeed multiple times since the neighborhoods overlap. Localization, on the other hand, is done on-line and therefore sampling of the current image is *sparse* at test-time.

2.2. Detection and Localization

During testing, *one* image at time τ can be tested against the template by *projecting it* as follows. Recall that each row i of the template $\phi \in \mathbb{Z}^{121 \times 16}$ is associated with one of the 121 *centers* c_i, and each column j measures the frequency of occurrence of either the intensity I_j in the histogram h^I or the gradient direction v_j in the histogram h^g. For each putative window location (x, y), we then sample, uniformly, *one* point for each neighborhood around each center $c_i(x, y)$, *i.e.* $(x_i, y_i) \sim \mathcal{U}(\mathcal{B}_r(c_i(x, y)))$, and read out the intensity $I_\tau(x_i, y_i)$ and the gradient orientation at the native scale, $\frac{\nabla I_\tau(x_i, y_i)}{\|\nabla I_\tau\|}$. Once we obtain 242 readings, we can quantize each of them into $K = 8$ intensity bins I_j and 8 direction bins v_j, and represent the histograms for a putative window location (x, y) as a 121×16 matrix:

$$J_\tau(x, y) = \{h^I_\tau, h^g_\tau\}. \tag{6}$$

Here, the histograms h^I_τ and h^g_τ are defined as in eq. (3) and eq. (2) for a test image at time τ, and are computed using *one* sample for each. We can then compare these histograms of $J_\tau(x, y)$ against the learned template ϕ using the entry-wise product:

$$\langle \phi, J_\tau(x, y) \rangle \doteq \|\phi \circ J_\tau(x, y)\| \tag{7}$$

where the norm is the sum of non-zero elements, and the argument of the norm is the entrywise product of the matrix ϕ and $J_\tau(x, y)$. We can also efficiently implement eq. (7) using bit-wise AND operations with SSE instructions similarly to [20, 8].

This projection represents the (un-normalized) P-value of standard hypothesis testing, and quantifies the likelihood of the window $w(x, y)$ at time τ under the hypothesis associated with the learned template ϕ. In practice, for computational speed as well as ease of normalization, it is easier to use the complementary score $2N_cM_c - \langle \phi, J_\tau(x, y) \rangle \geq 0$, which should be minimized for all possible choices of putative window location (x, y):

$$(\hat{x}(\tau), \hat{y}(\tau)) = \arg \min_{(x,y) \in \mathcal{X}} 2N_cM_c - \langle \phi, J_\tau(x, y) \rangle \tag{8}$$

where $\mathcal{X} \subseteq D = \mathbb{Z}^{N \times M}$ is a subset of the image domain. If we consider the entire image domain $\mathcal{X} = D$, the minimization procedure above corresponds to marginalizing the position nuisance (x, y) with respect to a uniform prior. Matching a learned template to the entire image domain is shown in Fig. 2, where the scale changes in each row from $S_1 = 0, 1, 2, 3$ levels in the image pyramid.

In practice, the prior is not uniform, as we can assume that, if the object was detected in the previous instant, it will be with high probability in a neighborhood of $(\hat{x}(\tau - 1), \hat{y}(\tau - 1))$. Furthermore, we can use a simple saliency detector (e.g. the local extrema of a differential operator, or a corner detector such as [17]) to avoid testing all possible putative locations. The goal of the saliency detector is not so much to detect regions where the target object is likely located, but more to discard regions where the target object is likely absent. In other words, the saliency detector acts as a conservative classifier for target location, that can have a large number of false alarms, but designed to keep the number of missed detection as close as possible to zero. Also notice that a putative location $(x, y) \in \mathcal{X}$ could be matched with the center of the window $w(x, y)$, but also with any of the centers $c_i(x, y)$. Therefore, if $\#\mathcal{X} = L$, we have to perform $121 \times L$ tests in order to detect the target. To further reduce the combinatorial search, we consider *not* all centers c_i, but only a subset of those, that we call *anchor points*, α_i, where the same saliency detector is most active, in other words, where features are most frequently detected in the neighborhoods. If we call the saliency detector mechanism $\psi : \Omega \subset D \to \mathbb{Z}$, acting in a domain Ω, then we have that an anchor point is

$$\alpha_i = \{c_i \mid \psi(\mathcal{B}_r(c_i)) < \theta\} \tag{9}$$

for an arbitrary threshold $\theta > 0$. Alternatively, we can sort all centers by their score $\psi(\mathcal{B}_r(c_i))$ and choose the first

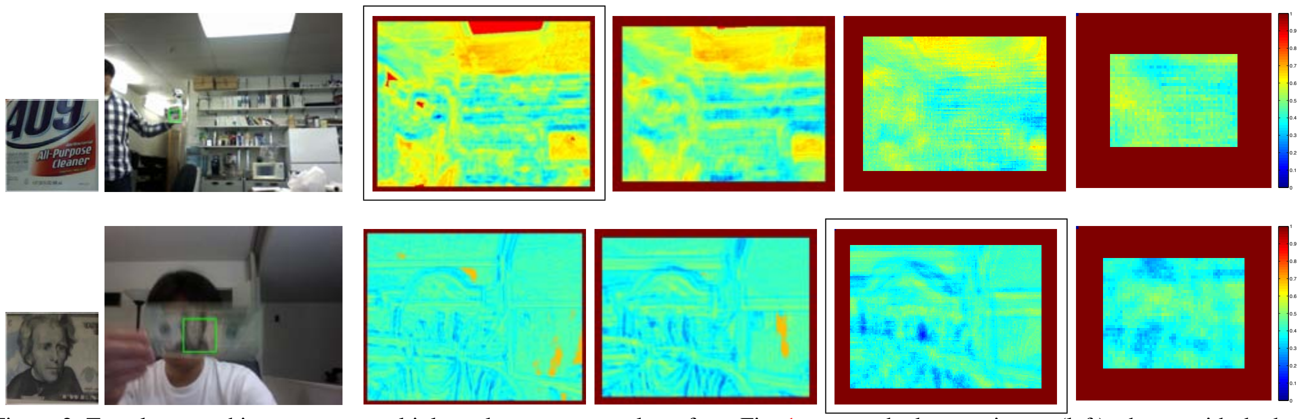

Figure 2. Template matching scores at multiple scales: target templates from Fig. 1 are matched to test image (left), shown with the best score match (green box). Multiscale template matching scores are shown with image pyramid levels $S_1 = 0, 1, 2, 3$. For each example, the best matches are found at the bordered levels.

A centers as anchor points. Thus, the set $\mathcal{X}$ is restricted to be a sparse collection of points in a neighborhood of $(\hat{x}(\tau-1), \hat{y}(\tau-1))$, and $A \times L$ evaluations must be performed at decision time. Fig. 1 shows an example of (left to right) one image from a training video sequence, the intensity histogram h^I, the gradient histogram h^g, the saliency of the centers and anchor points (yellow). Note, again, that this procedure is only meant as a computational expedient and is not strictly necessary, as one could just revert to the marginalization eq. (8) on the entire domain D. We evaluate the effectiveness of this anchor-based search in Section 4.

Although detection can be performed independently at each time instant τ, in practice one can exploit the presence of multiple images at test-time (live video) and use each instantaneous detector as a weak classifier in a cascade framework, where each subsequent detection reinforces or else reduces the current top hypotheses in eq. (8).

2.3. Tracking

Although recently localization has come to be accepted as a synonym of tracking (as in "tracking by detection"), we take tracking to imply a *temporal model* of the evolution of the object of interest. In the previous section we have assumed that the set $\mathcal{X}$ is centered at $(\hat{x}(\tau-1), \hat{y}(\tau-1))$, and therefore the next position of the object is equally likely in all directions. Such a model, whereby the expected position of the object of interest at time τ is centered at the object position at time $\tau-1$, does not qualify as a *bona fide* tracker in our parlance. However, a trivial n-order random walk is sufficient to enforce some rudimentary temporal regularity in the model, so we consider a simple robust Kalman filter (M-estimator) whereby the output (measurement) is provided by the detection/localization scheme described in the previous section, with the uncertainty equal to the (isotropic) standard deviation of the top m putative locations in the optimization eq. (8). A simple first-order random walk, with 4 states (position and velocity of the template) suffices to regularize the trajectories of the tracker, and enables pruning of the search space based on the predicted covariance. The pruning reduces the computation time for testing a template against the image as described in Section 4.

3. Rationale for the method

The method employed is described procedurally in the previous section, without justification for the choices made. In this section we provide our motivation for such choices following [18]. The reader who finds that performance in empirical tests is sufficient justification can skip ahead to the next section.

Object detection and localization are dual facets of the same problem, whereby one wishes to classify an image as either containing the object of interest λ or not (detection, $\lambda \in \{1, 0\}$), but the detection discriminant $p(\lambda|I_\tau)$ has to be marginalized with respect to location, (x, y). When the location prior is uniform, eq. (8) provides the marginalization mechanism, that returns the object location $(\hat{x}(\tau), \hat{y}(\tau))$ as a side effect.

What makes this problem difficult is the fact that the object of interest can undergo significant variability between the test and the training data, *even when it is the same object* and there is no intrinsic intra-individual variability within the category λ. Nuisance variability includes vantage point, illumination changes, occlusions, and quantization and other non-modeled phenomena. Nuisance variability should, ideally, be marginalized, and indeed we have done so for translation along the (x, y)-image plane. Translation along the optical axis can be approximated by scale changes σ, as in eq. (1). Unlike translation, we do not sample and search multiple scales as in eq. (8), but instead aggregate them into the descriptor ϕ. However, the projection eq. (7) can be effectively thought of as a "search" over scales, since it will yield a high P-value so long as the data sampled in J_τ matches at least one of the modes of the distribution implicit in ϕ. Rotation can be treated differently depending on whether the objects of interest are geo-located (e.g. a building), and therefore appear consistently in an inertial frame, or whether they are *detachable* (e.g., a book), so they can appear in any orientation. In the former case, translation can be easily *canonized* by referring it to a consistent reference, e.g., the direction of gravity, as most hand-held computing devices these days have inertial sensors available. For the case of detached objects, rotation can be canonized by choosing orientation statistics such as the maximum gradient direction in the window $w(x, y)$. This mechanism is, however, not very reliable, so some [9, 7] have suggested *not* canonizing rotation, but instead *averaging* the template across multiple rotations during learning eq. (5), using cross-validation to test the discriminative power of the resulting template, initiating a new template (with a different orientation reference) when discriminative power wanes. This can be done on-line, by incrementally performing the temporal averaging eq. (5), and using the template learned up to time T to classify the next data I_{T+1}, computing the leave-one-out score, and when this falls below a threshold, initialize a new template as if the object, oriented differently, was an entirely new object.

Occlusions can cause part of the window w to not be visible in the test image. This is accounted for in our method by the fact that the projection eq. (7) returns a high score so long as *some* of the histogram components, corresponding to *some* of the regions around the centers c_i, match the template. Occlusions are the very reason why we need to break down the window w into neighborhoods $\mathcal{B}_r(c_i)$.

Changes of illumination are difficult to model, but simple, coarse illumination changes can be represented as contrast (monotonic, continuous) transformation of the range of the image in the window region. It is well known that the gradient direction is a complete contrast invariant, so h^g captures all that matters of the window except for the effects of contrast transformations. On the other hand, although normalized intensity is robust to contrast changes, when occlusions occur, the intensity template h^I is can vary substantially. Thus, we use both the gradient orientation and the normalized intensity histograms despite their redundancy.

What remains unmodeled are the fine-scale deformations due to the interplay between changes of the vantage point and non-fronto-planarity of the scene: When we move in front of a non-planar object, the resulting deformation of the image domain can be quite complex and difficult to marginalize. For this reason, we do not marginalize it, and instead *"average it"* in the descriptor process eq. (5) in the same manner as we did for rotation.

It should be mentioned that the "averaging" of nuisances results, in general, in a loss of discriminative power with respect to marginalizing the discriminant, that however is costly at decision time. It is, however, the most common approach to deal with complex nuisances, as most existing "local descriptors" canonize simple transformations (e.g. similarity [13] or affine [16]), and then "blur" everything else by various binning or averaging mechanisms [2, 13, 19]. However, the advantage of eq. (5) is that, while all single-frame descriptors perform the average with respect to an ad-hoc prior (implicit in the histogram binning or blurring choices), the average in eq. (5) is performed with respect to *samples of the actual nuisance distribution*, since we are using images of the actual object obtained during training. In other words, if we write the descriptor ϕ_t explicitly as a function of the image I_t, $\phi_t(I_t)$, and the nuisance variability ν_t, which includes planar translation $(x(t), y(t))$, scale $\sigma(t)$ (equivalent to translation along the optical axis), rotation, contrast, partial occlusions, etc., so that $I_t \circ \nu_t \doteq \chi_t h_t(I_t(\frac{x-x(t)}{\sigma(t)}, \frac{y-y(t)}{\sigma(t)}))$, where h_t is a contrast transformation and χ_t the indicator function of the visible portion of the image domain, then we have that eq. (5) is $\phi = \frac{1}{T}\sum_t \phi_t(I_t \circ \nu_t)$ and is equivalent to a Monte Carlo evaluation of the integral:

$$\phi = \sum_{\substack{I_t \sim dP(I|\lambda) \\ \nu_t \sim dP(\nu)}} \phi_t(I_t \circ \nu_t) \simeq \int \phi_t(I \circ \nu) dP(I|\lambda) dP(\nu). \quad (10)$$

This is important, since the amount of "blur" is then object dependent: For a planar object, there will be little or no blur if we canonize affine transformations. On the other hand, for objects with complex shape, there will be significant blurring, provided that the training sequence is sufficiently complex. An early example of feature descriptor aggregated over multiple views during training is [15]. However, note that in general the discriminant built using the "blurred template" ϕ, where the nuisances are averaged in pre-processing, is not equivalent to proper marginalization of the nuisances, whereby the nuisances are integrated out at decision time. However, it is a reasonable compromise in our application context where one wants to minimize time-complexity.

4. Experiments

We evaluated the methods described in the previous section with respect to detection accuracy and computational efficiency. The dataset used for experimentation is composed of target objects captured at varying scale. Fig. 3 shows representative images of the dataset. The images were captured at 640×480 resolution, and target objects were trained from the centered 88×88 initial window tracked across multiple frames (i.e., 10 frames in our exper-

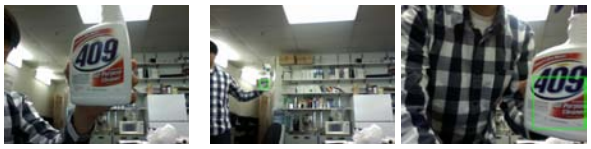

Figure 3. Example images from the dataset. Training a target object (left), and testing images with various sized targets (middle and right).

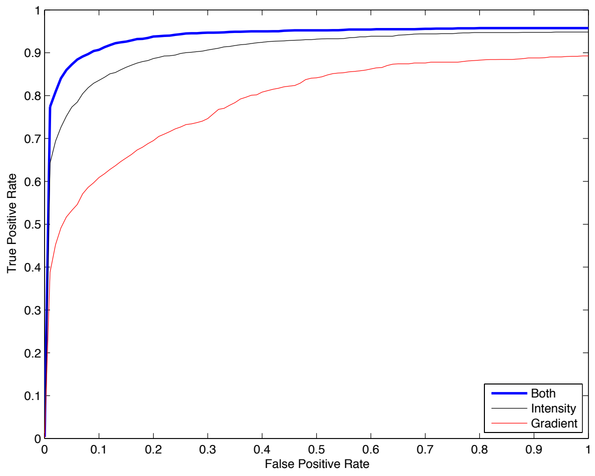

Figure 4. ROC curves from different types of template descriptors using intensity (black), gradient orientation (red), and both (blue).

iments). For each trained object, approximately 200 frames from the rest of the image sequences were tested. The test frames contain much more nuisance variability than the training frames, thus the low-level tracker in [22] fails systematically. The targets were typically seen in the image as small as 20×20, and as large as 400×400 pixels. Computational times are measured on a laptop with a 2.53GHz Core 2 Duo CPU.

To detect a target in the test image, a template descriptor is compared against *sample windows* by varying location and scale. We build an image pyramid with $S_1 = 5$ levels by blurring and downsampling by 2. Then we detect feature points using [17] at each level. The scale difference between the image pyramid levels is further divided by S_2 steps. For example, when $S_2 = 4$, the size of the sample window divides the scale range $[1..2]$ by $\{1, 1.25, 1.5, 1.75\}$ times the base scale. Thus, when $S_1 = 5$ and $S_2 = 4$, we have $S_1 \times S_2 = 20$ sample windows at different scales. The sample window is translated by assigning one of the anchor points to the feature points, and then tested with the template descriptor. The window that has the matching score above a threshold is selected as a target detection. Matched templates with position error less than 30% of the template width are considered correct.

First, we tested the combination of the intensity template and the gradient orientation template. Fig. 4 shows the ROC curves when using each template separately or together. Clearly, the joint template descriptor outperforms individual templates when they are used separately. Although the gradient orientation has lower accuracy than intensity, when they are used together, overall performance improves.

Next, we tested how the sampling scale steps S_2 improve the performance. Fig. 5 shows the ROC curves when using different number of steps for sampling scales per image pyramid levels. Using more than one step per each level significantly improves the performance. The difference among $S_2 = 2, 3, 4$ is marginal, where we can trade-off between accuracy and computational efficiency. From our experiments, $S_2 = 3$ was the chosen compromise.

We also tested the effect of different numbers of anchors in the template descriptor. Fig. 6 shows the ROC curves when varying the number of anchor points from 1 to 4. Us-

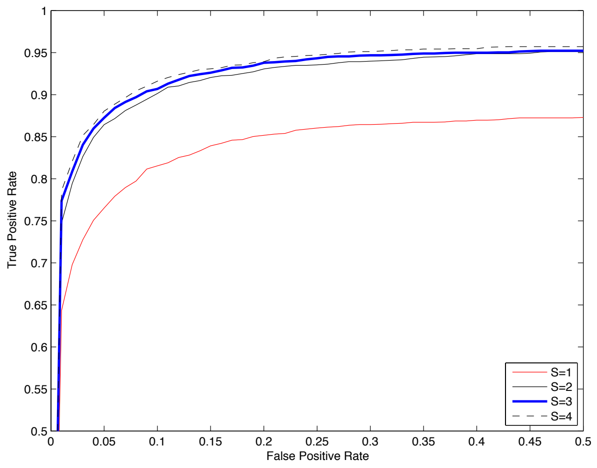

Figure 5. ROC curves with different number of sampling scale steps per image pyramid level. $S_2 = \{1, 2, 3, 4\}$.

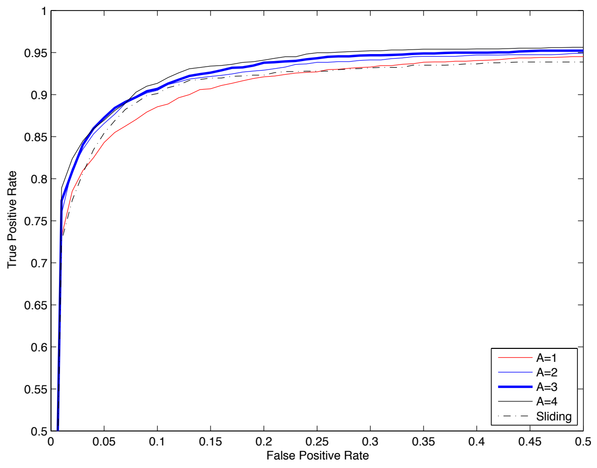

Figure 6. ROC curves with different window search methods using anchors (solid lines) and sliding windows (dashed line). Number of anchors = 4(black), 3(thick blue), 2(blue) and 1(red).

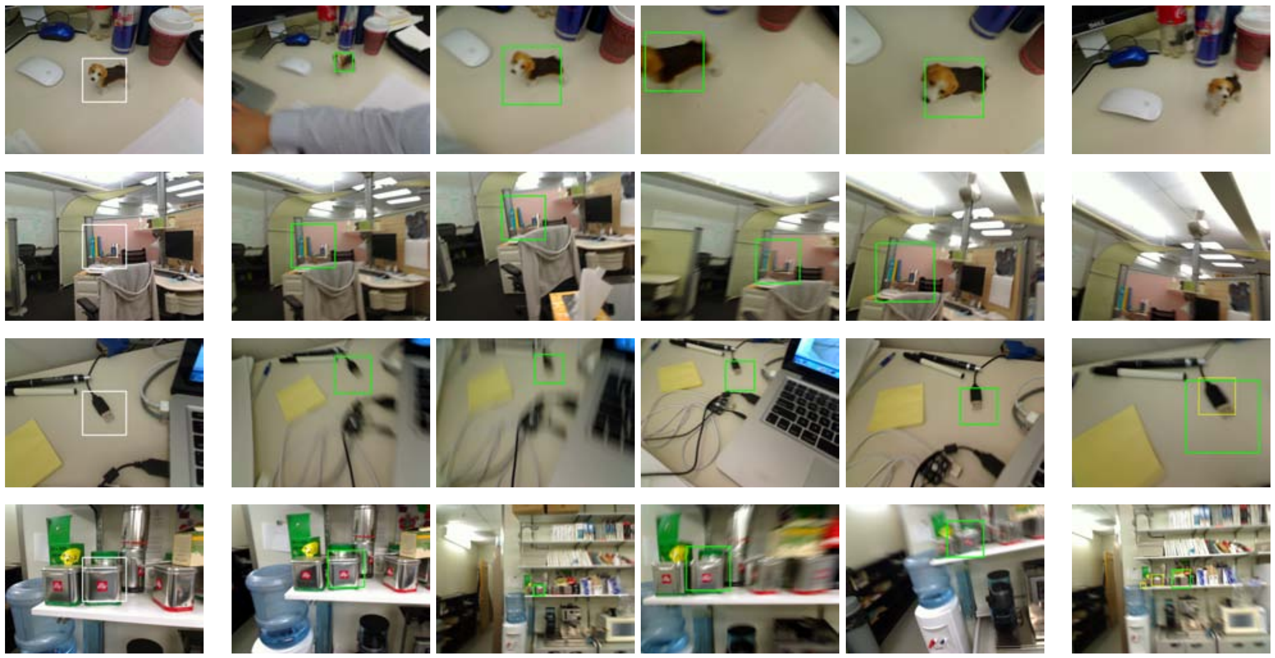
Figure 8. Example snapshots during detection and tracking of objects at multiple scales: (left column) target objects, (middle) successfully detected and localized targets with green bounding boxes. It is noticeable that the targets are detected under motion blur, viewpoint changes and scale changes. Unsuccessful matches are also shown due to incorrect estimates of scale and missing targets (right).

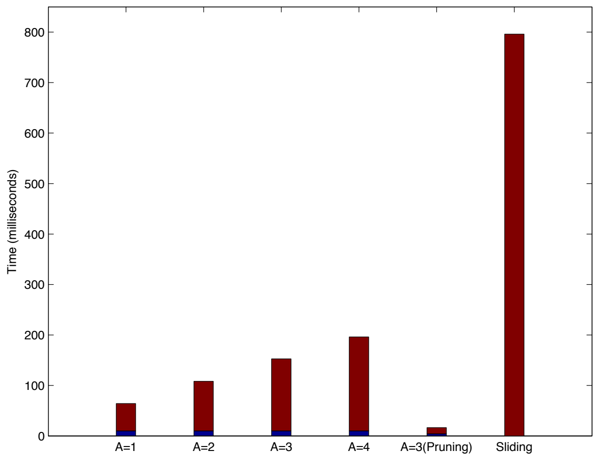

Figure 7. Average computation times for anchor-based search (left) and sliding-window search (right). The anchor-based search includes feature detection time. As the number of anchors increases, the computation time takes longer. $A = 3$ anchors with pruning has improves speed, compared to the sliding window search.

ing more than one anchor helps detection performance, because of the repeatability of the feature detection at multiple scales. In addition to the anchor-based search, the ROC curve of using the sliding window search is shown. Here, we slide the window by 20% of the window size. It is shown that the anchor-based search with more than one anchor point performs better than the sliding window search. Similar to the effect of varying the sample scale steps in Fig. 5, the difference among $2, 3, 4$ anchor points is marginal. For our experiments, using 3 anchor points is a good trade-off between accuracy and efficiency.

In addition, the computation time of the anchor-based search and the sliding window search was measured as shown in Fig. 7. As the number of anchors increases, the computation time increases linearly. When using 3 anchor points, it is about twice as fast compared to the sliding window search. Note again that searching with 3 anchor points is more accurate than the sliding window, as shown in Fig. 6.

The effect of anchor-based search is also tested for tracking a target object for multiple video frames. As in Fig. 7, using 3 anchor points with pruning the search space by the use of previous frame information helped the tracking speed. The result of tracking target objects are shown as representative snapshots in Fig. 8. The target objects were detected at multiple scales, and effectively tracked over image sequences.

In our experiments, we only learned one template in the beginning of each image sequence. Therefore it is expected that the detection accuracy decreases when the variation of the nuisances trumps the discriminative power of one template descriptor. In order to cope with this, we can adopt an online-learning mechanism as described in [8, 9] that initiates a new template when needed. However, we noted that reducing the number of templates per object instance is also important because of scalability in memory and speed. Combining the improved template descriptor with an efficient online learning algorithm will benefit the overall detection and tracking performance.

As for the feature detector or saliency detector mechanism, as used in our approach, the repeatability of the feature response is important. Because the anchor points rely on the probability that the features will be frequently detected again while testing a window at images. We chose [17] for our implementation because of its efficiency and reasonable repeatability, however, any similar feature detector algorithm such as [14] with improved computational efficiency and higher repeatability would enhance the anchor-based search method.

5. Discussion

We have described a method to perform detection and tracking of an object seen in a training video sequence, despite significant scale variations and partial occlusions. Matching in live video is expedited by the use of salient components of the descriptor (through the use of anchor points) and salient locations on a new image (through the use of a saliency mechanism, such as a "corner detector"). The method has been implemented and tested on live video, and representative qualitative and quantitative experiments have been reported. While no guarantees can be made on performance bounds, we have motivated our design choices based on assumptions underlying our model. These include Lambertian reflection, static illumination, and a modest to moderate amount of occlusions.

Acknowledgement

T. Lee was supported by Qualcomm's QinF Fellowship; Part of this research was supported by ARO 56765-CI and AFOSR FA9550-09-1-0427.

References

[1] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *Proc. of IEEE CVPR*, 2009. 1457

[2] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Proc. of ECCV*, pages 404–417, 2006. 1461

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/, 2009. 1457

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1459

[5] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. of IEEE CVPR*, pages 260–267, 2006. 1457

[6] N. Hagbi, O. Bergig, J. El-Sana, and M. Billinghurst. Shape recognition and pose estimation for mobile augmented reality. In *Proc. IEEE Int. Symposium on Mixed and Augmented Reality*, pages 65–71, 2009. 1458

[7] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit. Real-time learning of accurate patch rectification. In *Proc. of IEEE CVPR*, 2009. 1457, 1461

[8] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *Proc. of IEEE CVPR*, 2010. 1457, 1458, 1459, 1463

[9] Z. Kalal, J. Matas, and K. Mikolajczyk. P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. In *Proc. of IEEE CVPR*, 2010. 1457, 1461, 1463

[10] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proc. of IEEE CVPR*, 2008. 1458

[11] T. Lee and S. Soatto. TST/BTD: An end-to-end visual recognition system. *UCLA Technical Report, UCLA-CSD-100008*, February 10, 2010, revised March 18, 2010; http://www.youtube.com/watch?v=cMv-McHw660. 1459

[12] V. Lepetit and P. Fua. Keypoint Recognition using Randomized Trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479, 2006. 1457

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2(60):91–110, 2004. 1458, 1461

[14] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proc. of ECCV*, September 2010. 1464

[15] J. Meltzer, M. Yang, R. Gupta, and S. Soatto. Multiple view feature descriptors from image sequences via kernel principal component analysis. In *Proc. of ECCV*, pages 215–227, May 2004. 1461

[16] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. of ECCV*, pages 128–142. Springer-Verlag, 2002. 1461

[17] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proc. of ECCV*, volume 1, pages 430–443, May 2006. 1459, 1462, 1464

[18] S. Soatto. Steps toward a theory of visual information. *UCLA Technical Report, UCLA-CSD-100028*, September 13, 2010. 1460

[19] G. Takacs, V. Chandrasekhar, S. Tsai, D. Chen, R. Grzeszczuk, and B. Girod. Unified real-time tracking and recognition with rotation-invariant fast features. In *Proc. of IEEE CVPR*, pages 934 –941, June 2010. 1461

[20] S. Taylor and T. Drummond. Multiple target localisation at over 100 fps. In *Proc. of BMVC*, September 2009. 1457, 1459

[21] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368, 2010. 1458

[22] Willow Garage. OpenCV: Open source computer vision library. http://opencv.willowgarage.com/wiki/, 2009. 1462