Articulated Object Tracking By Rendering Consistent Appearance Parts

Zachary Pezzementi Sandrine Voros Gregory D. Hager

***Abstract*— We describe a general methodology for tracking 3-dimensional objects in monocular and stereo video that makes use of GPU-accelerated filtering and rendering in combination with machine learning techniques. The method operates on targets consisting of kinematic chains with known geometry. The tracked target is divided into one or more areas of consistent appearance. The appearance of each area is represented by a classifier trained to assign a class-conditional probability to image feature vectors. A search is then performed on the configuration space of the target to find the maximum likelihood configuration. In the search, candidate hypotheses are evaluated by rendering a 3D model of the target object and measuring its consistency with the class probability map. The method is demonstrated for tool tracking on videos from two surgical domains, as well as in a human hand-tracking task.**

I. Introduction

There are many applications in which the fundamental problem is to track an articulated object with changing appearance in a changing background. Examples range from tracking rigid objects in motion [15] to tracking robot arms or hands [30], [6] to tracking one or more humans in images [10]. In almost all cases, it is reasonable to assume that a reasonably accurate kinematic model is available for the target in question. However, what is not known, a priori, is the appearance of the target and, more importantly, what distinguishes the target most effectively from the background.

The problem of simultaneously learning and tracking a model has seen a great deal of activity in recent years. Early work developed per-pixel generative models for the foreground only [25] (resp. background only [33]). Background (resp. foreground) objects were then detected as "outliers" to the process. More recent work such as [37] develops both foreground and background appearance models using spatially augmented GMMS. In work specific to tracking, [1] and [12] develop and maintain an ensemble of weak classifiers over time by re-weighting, adding or rejecting classifiers. Lu [24] maintains a "bags of image patches" appearance model using a temporal-adaptive importance resampling procedure. In general, in these approaches a classifier is trained over a set of images using selected features (for instance color histograms in [38] or a combination of local orientation histogram and pixel colors in [1]), to identify and discriminate foreground from background.

In the work cited above, the tracked target has no a priori geometric structure, and tracking amounts to maintaining some notion of location or rough target outline over time. Often, there is a need for information beyond location. Early work in human tracking made use of local features such as edges [15], [10] to optimize over body pose, building upon several systems for tracking rigid or low-DOF articulated 3D objects which rely upon visual contours [9], [29], [22], [28]. Although this may work for models with well-defined edges before uncluttered backgrounds, these contours can not be reliably detected in many real-world settings. For instance, the surgical setting from which most of our examples are drawn is rife with features which are detected as spurious edges by all standard edge detectors, as seen in Figure 1, and which would confuse any solely edge-based approach. Dealing with so many outliers would likely be very expensive. These same challenging visual effects, specularities and inconsistent lighting, occlusion, and small depth of field, pose problems for other local features as well. We opt, instead, to take a more "image-based" approach, making use of all visible parts of the target object, rather than only those which are most distinctive.

Also in the human tracking field, other local features such as optical flow [21], template-matching [3] or object texture [2] have been employed. Other work has explored interest point correspondences [13], [35]. The 3D pose recovery is then either formulated as an inverse kinematics problem or a constrained optimization problem, the latter formulation typically being more robust over time [10]. Recent work on human tracking [31] uses conditional Bayesian mixtures of kernel-induced experts to build "bottom-up" discriminative models of pose probabilities from features taken from the silhouette of the target. In [27], the authors detect a model of parts consisting of parallel edges, then learn appearance models from those detections to use for tracking within the sequence. However, in most of this work, the focus is on detection, and tracking is treated as a "sequential detection problem" rather than an optimization over the configuration space of the target.

Thus, despite this volume of work, it is unfortunately still the case that there are few general-purpose tools for robust and efficient tracking of kinematic structures with fully or approximately known geometry and unknown appearance. Our work aims to create such a tool by combining statistical learning methods to create a generative model of appearance similar to [6], [24], [32], [38] with a structured, kinematically-based optimization to best align the geometric model with the data as [11] or [2]. Since the geometry of the model is known, one can view this problem as one of projecting the model into one or more images, then associating each projected part with a model of approximately uniform appearance, using image information. As we will show, our algorithms rely on relatively simple

Laboratory for Computational Science and Robotics (LCSR), Johns Hopkins University, 3400 North Charles Street, Baltimore, MD 21218 {zap, svoros, hager}@cs.jhu.edu

image filtering, 3D projection, and appearance evaluation algorithms. As a result, most calculations can be performed by commodity GPU hardware, making the achievement of real-time performance relatively straightforward.

II. Tracking from Rendered Maps

Our tracking method consists of two components: 1) an appearance modeling step that computes a probability of class membership for each appearance class and for each pixel, and 2) an optimization step or filtering step that computes the configuration of the object from this probability map. Here, we present the underpinnings of the latter and defer our discussion of appearance classification to Section III-A.

As a starting point, we note that any serial chain mechanism consisting of rigid bodies can be represented by a minimal set of variables that in turn define the object's configuration space [23]. In what follows, let $\mathbf{x}$ denote some parameterization of the configuration of an object. We denote an image by $\mathbf{I}$ and, pixel-related feature vectors in the image by $v \in \mathbf{I}$. Let $\mathcal{C}$ denote a parameterization of an appearance model on image feature vectors and let $\theta_f = \{\mathcal{C}_1, \mathcal{C}_2, \ldots \mathcal{C}_m\}$ represent the set of appearance models for a "foreground" object in question. We denote the background appearance class by $\mathcal{B}$ and define $\theta = \theta_f \cup \{\mathcal{B}\}$. We assume that, for each appearance class in question, we can "render" the area of the target into the image to create a mask which in turn identifies a set of these feature vectors. We write $\mathcal{C}(\mathbf{x})$ (resp. $\mathcal{B}(\mathbf{x})$) to denote the set of pixel vectors in the image corresponding to the mask generated for a given class and an object configuration. These form a strict partition on all feature vectors in an image. Finally, we distinguish between estimated quantities and random variables by writing them with a "hat" over them, i.e. $\hat{x}$ vs. x.

There are a variety of estimation-theoretic paradigms for posing and solving the problem of configuration estimation. Two of immediate interest are 1) maximum likelihood estimation and 2) time-series filters that rely on importance sampling. Both methods rely on the ability to evaluate the likelihood function at a particular object configuration. Assuming independence of feature vectors given a classification, the likelihood function can be written as

$$P(\mathbf{I} \mid \mathbf{x}, \theta) = \prod_{\mathcal{C} \in \{\theta\}} \prod_{v \in \mathcal{C}(\mathbf{x})} P(v \mid \mathcal{C}) \tag{1}$$

Define $L(v, \mathcal{C}) = -\log P(v \mid \mathcal{C})$. Taking the negative log of (1), we have

$$-\log P(\mathbf{I} \mid \mathbf{x}, \theta) = \sum_{\mathcal{C} \in \theta} \sum_{v \in \mathcal{C}(\mathbf{x})} L(v \mid \mathcal{C}) \tag{2}$$

$$= \sum_{\mathcal{C} \in \theta_f} \sum_{v \in \mathcal{C}(\mathbf{x})} L(v \mid \mathcal{C}) + \sum_{v \in \mathbf{I}} L(v \mid \mathcal{B})$$
$$- \sum_{\mathcal{C} \in \theta_f} \sum_{v \in \mathcal{C}(\mathbf{x})} L(v \mid \mathcal{B}) \tag{3}$$

$$= c + \sum_{\mathcal{C} \in \theta_f} \sum_{v \in \mathcal{C}(\mathbf{x})} \left(L(v \mid \mathcal{C}) - L(v \mid \mathcal{B})\right) \tag{4}$$

where c is a constant factor that varies from image to image, but is independent of $\mathbf{x}$ within a given image.

As a consequence, we can now write (1) in an equivalent form as

$$P(\mathbf{I} \mid \mathbf{x}, \theta) = \kappa \prod_{\mathcal{C} \in \theta_f} \prod_{v \in \mathcal{C}(\mathbf{x})} \frac{P(v \mid \mathcal{C})}{P(v \mid \mathcal{B})} \tag{5}$$

It is interesting to note that, for a single foreground class, the resulting expression takes the form of a likelihood ratio.

One way of performing tracking is to simply compute the maximum likelihood pose at each step, i.e. to solve

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} P(\mathbf{I} \mid \mathbf{x}, \theta) \tag{6}$$

$$= \arg\min_{\mathbf{x}} \sum_{\mathcal{C} \in \theta_f} \sum_{v \in \mathcal{C}(\mathbf{x})} L(v \mid \mathcal{C}) - L(v \mid \mathcal{B}) \tag{7}$$

In general, this optimization can be quite complex, as the search space can be high-dimensional and the objective function is not guaranteed to be convex. Our current methods for solving it are described in Section III. However, for special cases, the solution is immediate. For example, when computing image location (i.e. object translation when motion lies approximately in a plane) with a single foreground class we can consider the mask as a kernel. Then, (7) can be solved by finding the location of the global maximum when convolving the target mask over the difference between foreground and background classification log probabilities. Local optimization can be easily accomplished using the mean shift algorithm [4]. Indeed, exactly this approach is applied in [24].

Finally, we note that, given a method for sampling the likelihood function, methods for developing a particle filter for tracking configuration are well-established [8].

III. Implementation

The previous section provides a very general framework for the design of articulated tracking design from learned appearance models. We now turn to the specific choices made in our current implementation. We first describe our appearance modeling Section III-A, then discuss rendering and optimization details in Section III-B.

A. Appearance Learning

Our appearance model computes class-conditional probabilities from color and texture features extracted from the image, as described in the applications in Section IV. We currently assume at least one labeled image of the target is available. Using the labeled image, we apply linear discriminant analysis (LDA) to compute a linear dimensional reduction using the most discriminating features. Then, one of many possible statistical learning techniques is used to train a model capable of producing class probability estimates. In the example applications that follow in Section IV, both histograms and Gaussian mixture models are used, though the field of machine learning provides a wealth of other possibilities.

The initial model training is currently done by hand by aligning or hand-segmenting a model to one or more images

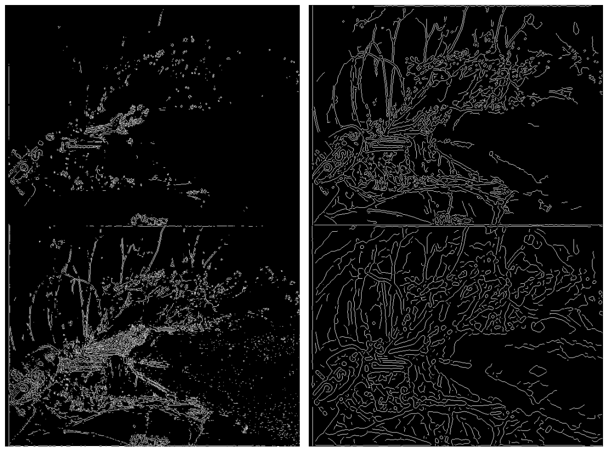

Fig. 1. Typical edge detection results on an image from the laparoscopic surgery domain. The input is the second image from Figure 2. On the left are two parameter settings for the Sobel filter and on the right two settings for the Canny detector. In all cases, the tool shaft is not well-detected, and many edges are detected in the background in the vicinity of the tool tip.

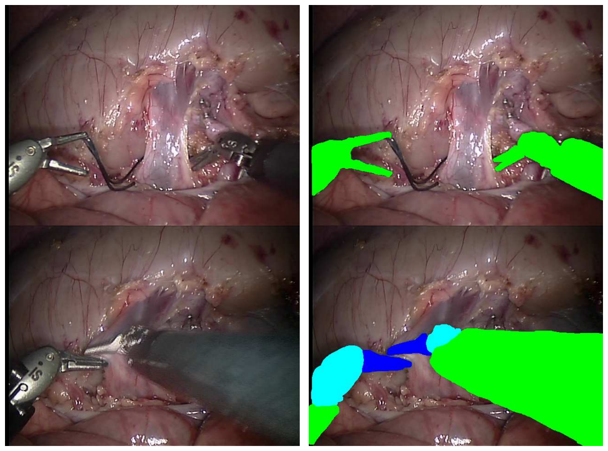

Fig. 2. Hand-segmented images from the da Vinci laparoscopic surgery domain used to train appearance models. In the first row, a single foreground class is used and is labelled green. The second row shows 3 foreground classes: green is tool shaft, cyan is wrist, and blue is finger.

of the data set, as shown in Figure 2. As we discuss in Section V, more automated training is possible, but we have not fully developed this aspect of the method yet.

In the results presented in this paper, we have found it sufficient to train models on one or two images in the sequence without further adaptation. Preliminary tests did not show a large variation in the performance of different classifiers on this task.

B. Rendering and Optimization Details

In our implementation, the image mask is evaluated by standard graphical rendering techniques given a 3-dimensional model of the object: the object models were created in Pro/ENGINEER from measurements of the objects and exported as Wavefront triangular meshes. Articulated objects were modeled by part and then configured at render-time. Rendering was performed using OpenGL to take advan-

Operation	Time/Op (ms)	Num. Ops.	Total (ms)
11x11 filter	3.8	4	15.2
3x3 filter	0.16	4	0.64
Probability	1	1	1
Render	0.1	max	max
Image Mask	0.1	max	max
Image Sum	0.3	max	max

TABLE I

THE APPROXIMATE TIME REQUIRED TO PERFORM WHOLE-IMAGE (640-X-480) OPERATIONS ON OUR TEST SYSTEM GPU (GEFORCE 8600GTS). PROBABILITY REFERS TO COMPUTING A CLASS PROBABILITY MAP AND CONSISTS OF AN LDA PROJECTION AND TABLE LOOKUPS. RENDER REFERS TO GENERATING AN IMAGE OF THE OBJECT MODEL. IMAGE MASK IS AN ELEMENT-WISE MULTIPLICATION OF A RENDERED IMAGE WITH A PROBABILITY MAP, AND IMAGE SUM IS SUMMATION OF ALL ELEMENTS IN AN IMAGE. NUM. OPS. GIVES THE NUMBER OF TIMES EACH OPERATION MUST BE PERFORMED ONA FRAME OF VIDEO, WITH "MAX" MEANING AS MANY AS POSSIBLE IN THE TIME AVAILABLE. TOTAL GIVES THE OVERALL AMOUNT OF TIME SPENT ON THAT OPERATION EACH FRAME.

tage of hardware-acceleration. For stereo video sequences, this step required a known camera calibration to establish the relationship between the 3D model and its projection on the image plane. Camera(s) calibration(s) were obtained using Matlab/OpenCV Calibration Toolbox [34].

The results in this paper are all generated using maximum likelihood estimation. In order to optimize (1), we elected to use the Nelder-Mead downhill simplex method [26] since it requires the computation of only function values, not derivatives, and is robust to noisy minimization functions. Since we are projecting the entire tool into the image, local optimization from the previous target configuration typically converges reliably, provided there is sufficient overlap from the previous frame. Given the relatively slow motion, and the relatively high dimensionality of the problem, we found that spatial importance sampling did not provide a noticeable improvement over MLE.

In addition to the use of hardware-based rendering, we have performed preliminary tests of the time required for each step of the algorithm using a GPU implementation of GLSL shaders [17]. Our current results are shown in Table I. The time required for applying whole-image filters of two sizes is given, as well as the number of such operations required for feature extraction. The timing of the generation of the probability map is also estimated, as well as the two parts of the image dot product that make up the evaluation of the objective function. The rows of Table II show the number of computations of the objective function expected to be possible at each frame-rate, based on those timings, with real-time performance expected in the final configuration, that of a high-end gaming system.

The hardware of the testing machine consisted of a dual-core 3-GHz Intel CPU, 2GB of RAM and an nVIDIA GeForce 8600GTS for graphics processing. Input images were 640-x-480 resolution. Runtime on each sequence was

Device	30 fps	15 fps	10 fps	5 fps	2 fps
8600GTS	33	100	166	366	966
8800GTX	233	380	584	1291	3515
2x8800GTX	500	913	1384	2891	7515

TABLE II

NUMBER OF COMPUTATIONS OF THE OBJECTIVE FUNCTION EXPECTED TO BE POSSIBLE AT EACH FRAME-RATE. EACH DEVICE IS AN NVIDIA GEFORCE MODEL. 8600GTS IS THE TEST MACHINE, AND THE OTHER TWO ARE EXTRAPOLATED VALUES BASED ON HARDWARE DIFFERENCES.

around 800ms for feature extraction and classification and 4 to 5 seconds for minimization in each frame. A realistic GPU implementation could make use of a higher-end graphics card such as a GeForce 8800GTX which has four times as many stream processors as the test system's card as well as additional video memory. The last two rows of Table II assume speed-up factors of 4 and 8 from substituting one or two of these cards respectively. Other optimizations, such as down-sampling of the input images or only classifying pixels in the neighborhood of the target, as well as hand-optimization of the operations presented above or implementation in CUDA [5], could yield significant further speed-ups. Real-time performance could therefore be achieved for relatively high-dimensional models using widely-available hardware.

In summary, the parameters that a user must set consist of the following: For appearance modeling, the number of appearance classes to use to represent the model must be chosen, as well as the number of components to use in the mixture model and the size of the feature extraction filters (which should be related to the size of input images). For optimization, assuming the use of downhill simplex, one must set the initial simplex and choose the termination conditions, i.e. convergeance criteria for absolute error or its derivative, or maximum iterations.

IV. RESULTS

We have applied the tracking method to a variety of monocular and stereo tracking problems. These include monocular tool tracking in microscopic sequences, tracking two articulated tools in stereoscopic video of robotic surgery, and monocular hand tracking. We also provide results from a simulated video data set. Each of the complete video sequences is available in the supplementary materials, as well as via web on the CIRL web-site.

On the simulated sequence, the ground truth configuration is known, and accuracy is measured by the magnitude of difference in the configuration estimate from the true configuration, $\frac{||\hat{\mathbf{x}}-\mathbf{x}||}{||\mathbf{x}||}$. Since the true configuration is not available for the real sequences, accuracy is instead estimated by the pixel overlap of the rendered tool model and a hand segmentation of the sample images, shown in Table III. Here, we use the information retrieval metrics recall, precision and the probability of error, PE [20], defined in terms of true positives, TP, true negatives, TN, false positives, FP, and false negatives, FN, as

Sequence	Precision	Recall	PE	Config
Retinal	0.708	0.850	0.007	-
Simulated	0.924	0.966	0.016	0.010
Lap-S	0.733	0.926	0.044	-
Lap-M	0.895	0.750	0.017	-
Hand	0.942	0.882	0.014	-

TABLE III

THE FIRST 3 COLUMNS SHOW AVERAGE IMAGE OVERLAP ACCURACY ON SAMPLED IMAGES FROM EACH VIDEO SEQUENCE. F INDICATES THE F-MEASURE ($F = 2(P \times R)/(P + R)$). CONFIG GIVES THE AVERAGE ERROR IN THE CONFIGURATION SPACE, NORMALIZED BY THE AVERAGE MAGNITUDE OF THE TRUE CONFIGURATION VECTOR, AND IS ONLY AVAILABLE FOR THE SIMULATED SEQUENCE. EACH MEASURE IS RANGED $[0, 1]$. LAP-S AND LAP-M ARE THE LAPAROSCOPIC SURGERY SEQUENCES WITH A SINGLE OR MULTIPLE FOREGROUND CLASSES RESPECTIVELY. IN THE MULTI-CLASS CASE, ALL PARTS OF THE RENDERED TOOL ARE CONSIDERED "POSITIVE" INSTANCES.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{8}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{9}$$

$$PE = (TP+FN)\,FN + (TN+FP)\,FP \tag{10}$$

where we consider pixels labelled as the target object to be positive examples (and background negative), and the hand segmentation to be the ground truth. The F-measure is intended to give a sense of "overall" accuracy by taking into account both precision and recall, since either one alone can be trivially driven to 1 by assigning everything to background or to foreground respectively.

A. Retinal surgery

We first present results of the algorithm performing rigid tool tracking on video from a retinal eye surgery test-bed. In this case, stereo microscope images of a pair of forceps against an eye phantom background are captured. The phantom is the interior of a half-sphere, painted to resemble the retinal surface.

The main challenge of this domain comes from the optics of the microscope. Due to the extreme magnification, the depth-of-field is very small, so it is impossible to keep the entire scene in focus. As a result, most of the tool shaft is blurred, making edge-based tracking challenging if not impossible. Additionally, this effect makes it difficult to get a good stereo calibration of the system. For this reason, tracking in this domain was performed on monocular video.

We perform tracking in this sequence using a 2-D mask taken from a hand-segmentation of a single frame of the video. The pose space consists of 2-D rotation and translation of this mask. Although the tool is not actually rigid, since the fingers of the forceps are articulated, we model it as such, and it remains mostly rigid within this video sequence. Sample images are shown in Figure 3.

In this case, the probabilities in Equation 1 are obtained from histogram matching. During training, a histogram $h_{\mathcal{C}}$ is built for each class, $\mathcal{C}$, in the LDA space in the usual fashion, and normalized to sum to 1. Then, during classification, the class probability of a pixel is taken as

$$P(v \mid \mathcal{C}) = h_{\mathcal{C}}\left(u(\pi v)\right) \tag{11}$$

where $u(v)$ is the histogram binning function and π is the linear dimensional reduction given by LDA. The features input to the LDA computation were RGB and HSV color of the pixel, average intensity within a small window, and 5 Laplacian of Gaussian filters of different bandwidths to add texture information. Finally, the minimum and maximum values over the vector comprising these features were added as two more features.

Given the simplicity of the problem, we were able to implement the tracking algorithm in Matlab, using an optimization which iteratively searched sucessive 2D rotations and translations of the mask. This allowed us to compare the tracking results using a single model and with those obtained when performing model updating on each frame. The result were nearly identical, suggesting the algorithm is reasonably robust to moderate changes in appearance.

B. *Laparoscopic surgery*

Here, we track two needle driver tools of Intuitive's da Vinci robotic system [16]. The tools consist of a long shaft, followed by a wrist joint, which is connected to two grippers that can move independently. We also allowed the base of the shaft to rotate and translate freely giving a total of 9 degrees of freedom per tool. We simultaneously track two of these tools to test performance on tracking articulated objects. Thus, the final configuration space consists of 18 degrees of freedom.

Challenges of this domain include extensive specular reflections, a cluttered and changing background, and tools which change appearance as they rotate or become covered with body fluids.

Here we chose to use Gaussin mixture models (GMMs) to represent the appearance of the tool along the lines of [19]. Both LDA and GMM computations are performed by LNKnet [18]. Once trained, the class-conditional probabilities needed for Equation 1 are evaluated as usual [36], as a function of the means, μ_i, covariances, Σ_i, and weighting coefficients, α_i, of the $\mathcal{C}^n$ mixture components of class $\mathcal{C}$, with $i \in \{1, 2, \ldots \mathcal{C}^n\}$, and with π once again representing the LDA transformation:

$$P(v \mid \mathcal{C}) = \sum_{i=1}^{\mathcal{C}^n} \alpha_i P(\pi v \mid \mu_i, \Sigma_i) \tag{12}$$

where

$$P(\pi v \mid \mu_i, \Sigma_i) = \frac{1}{\sqrt{2\pi}|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(\pi v - \mu_i)^T \Sigma_i^{-1} (\pi v - \mu_i)} \tag{13}$$

The particular features used are the average color value of a 3-by-3 block of pixels centered at the current location, represented in both RGB and a modified version of HSV, ConeHSV, which is defined with respect to the standard components of HSV (H, S, and V) as

$$ConeHSV_1 = V \tag{14}$$

$$ConeHSV_2 = S\cos(H) \tag{15}$$

$$ConeHSV_3 = S\sin(H) \tag{16}$$

The effect is to reshape the standard HSV space to the conical representation sometimes used for color pickers, in which Euclidean distances are more valid. To these six features, four Haralick texture features [14] are added (contrast, correlation, energy, and homogeneity), which are based on gray-level co-occurrence matrices built from a 7-x-7 neighborhood about the current location. The final feature vector is of total length 10 with each entry ranged 0 to 1. One could choose to use a set of features tailored to the specific domain in which tracking is being performed. For simplicity and generality, we use this same set of features in each tracking domain, as it provides two complementary color descriptors as well as a rich but compact set of texture features, although Gabor filters would probably be a more suitable texture feature choice for a GPU implementation.

1) Simulated Data: We first tested the algorithm on a straightforward simulated problem. A simulated data set was generated by rendering the da Vinci tool models undergoing some translation and articulation against a black background. This sequence effectively bypasses the class probability estimation step, giving the case of an ideal segmentation and perfect object model. The ground truth configuration of the object is also available in each frame. Error was evaluated by two metrics, average Euclidean distance in the configuration space of joint angles and translations, and the image overlap metric used for the real sequences. Sample images are shown in Figure 4. As can be seen in Table III, the final configuration error was 0.01 which is at the convergence resolution of the optimization.

2) Real Data: We conducted tracking trials on a stereo video sequence of a suturing task in porcine surgery. We tested two cases: 1) a single foreground class and 2) two foreground classes. Sample images of both are shown in Figure 5. In the latter case, the needle-driver was initially modeled as 3 separate classes: shaft, wrist, and fingers, as illustrated in Figure 2. Wrist and finger pixels were not reliably distinguished though, so they were merged into a single end-effector class used in the results shown.

One can see that the multi-class case displays more reliable tracking of the end-effector. This is not surprising, as the LDA reduction we currently use would have difficulty finding a single projection that distinguishes both from the background. The precision of the method was nearly as good as the simulated sequence; however the recall is much lower, mainly due to the tool being not completely detected in the class probability estimation step.

C. *Hand Tracking*

We also demonstrate a hand-tracking task on a short mono video of an arm moving against a static background, in

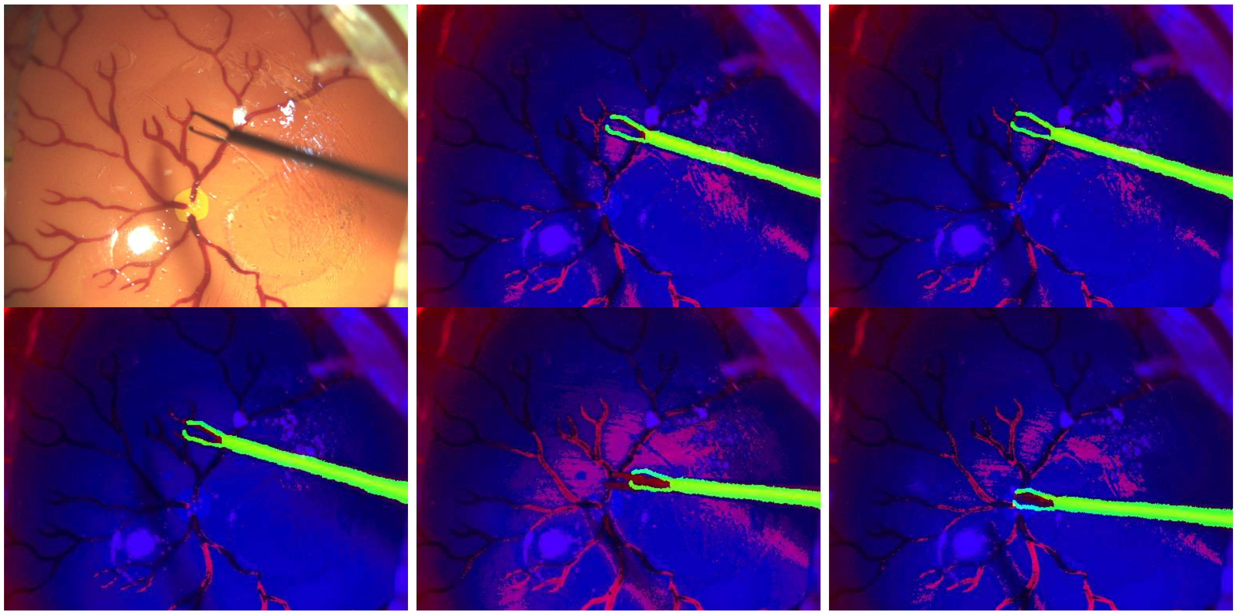

Fig. 3. Images from the retinal microsurgery domain, demonstrating rigid tool tracking: top left is the first input image of the video, then the tracking results for frames 1 and 17. On the second row are tracking results from frames 33, 49, and 65. For the tracking result images, the blue channel displays the input image intensity, the red channel shows the probability of tool output by the classifier, and the green channel shows the rendered tool model in the current belief configuration.

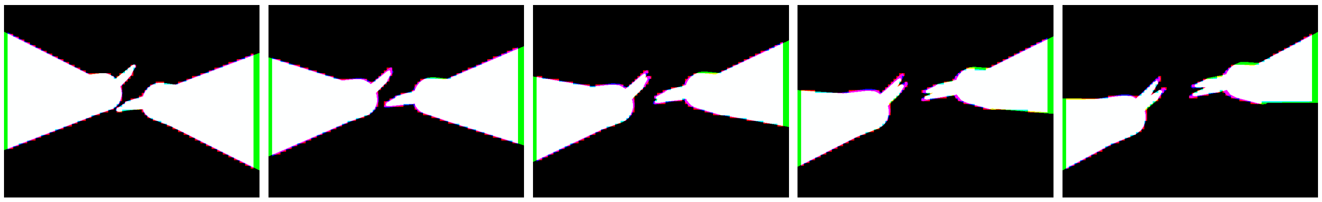

Fig. 4. Images from the simulated da Vinci laparoscopic surgery sequence: Frames 1, 25, 50, 75, and 100 of the left channel of the tracked sequence are shown. The color-scheme is the same as in Figure 3, though in this case the red and blue channels are identical.

order to show the versatility of the method across different domains. Our model is quite rudimentary: each finger is modeled as a rigid cylinder with rounded ends and two degrees of rotational freedom. The wrist also has two degrees of rotational freedom. With translation of the base of the forearm, there are a total of fifteen degrees of freedom in our model, much less than the estimated 27 of the real hand [7]. The same appearance model was used as in the laparoscopic surgery application in Section IV-B.

Sample images are shown in Figure 6. In particular we can see in frames 11 and 21 that the optimization appears to have chosen a local minimum that could be easily improved by slightly moving the thumb and forearm. Again, precision is relatively high, while recall is somewhat lower. In this case, this seems most likely due to inaccuracy in the 3D model at the edges causing a more conservative fit to be more optimal.

V. Discussion

The work we have presented here is a promising first step toward a general-purpose and efficient methodology for tracking articulated objects. What is particularly compelling is that, even without sophisticated search algorithms and time-series filtering, the algorithms are able to track highly articulated surgical tools in challenging real-world situations, and operate with an image-domain accuracy that is acceptable for many applications. Most importantly, the algorithms were applied with minimal changes in three different domains with very different articulated models.

There are several immediate extensions that are clearly needed to improve the accuracy and overall performance of the method. As noted in Section III-B, then entire evaluation of the likelihood function can be implemented on the GPU, providing several orders of magnitude reduction in execution time. This in turn will make it possible to examine a large family of sampling-based search algorithms, particularly stochastic gradient methods and particle filters, with very large numbers of samples.

The generation and use of the class conditional probability maps also has many potential improvements. Making use of boundary information on tools, and unique image features (potentially registered to the underlying CAD model) are likely to improve both accuracy and robustness.

Although an approach relying entirely on edge features is not likely to fare well in the laparoscopic surgery do-

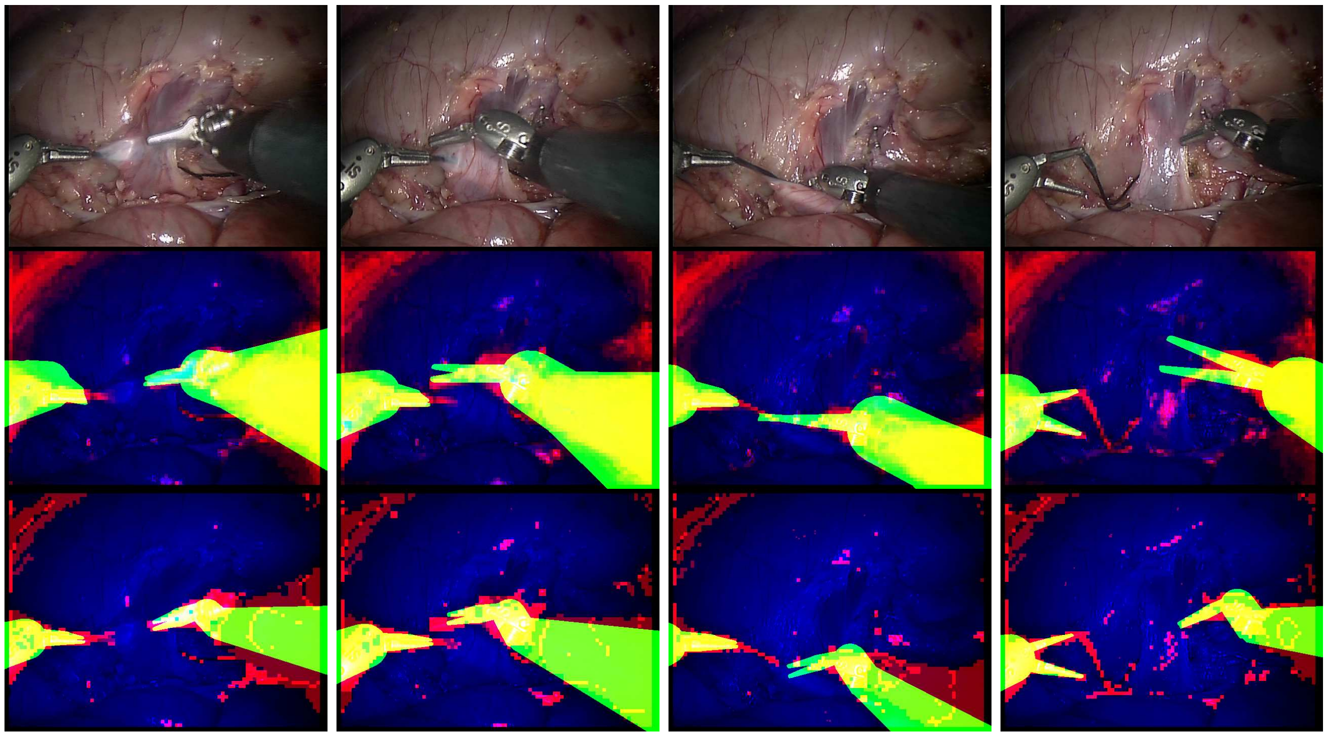

Fig. 5. Images from the da Vinci laparoscopic surgery domain, demonstrating articulated tool tracking: The top row shows input frames 25, 75, 125, and 175 from the left channel of video. The middle and bottom rows show the corresponding tracking output for a single foreground class and two foreground classes respectively. The color-scheme for the middle row is the same as in previous figures: the blue channel displays the input image intensity, the red channel shows the probability of tool, and the green channel shows the the tracked tool. In the bottom row, the red channel does not show foreground probability, but instead a value corresponding to the most probable class: zero for background, half intensity for tool shaft, and full intensity for tool end effector.

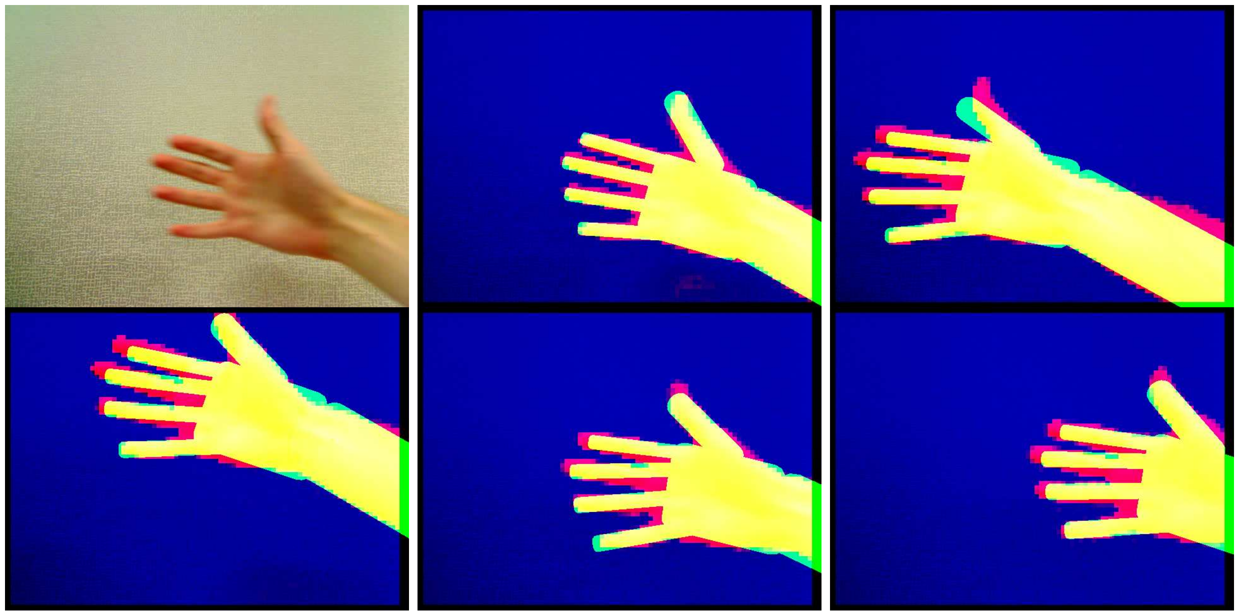

Fig. 6. Hand tracking results: top left is the first input image of the video, then the tracking results for frames 1 and 11. On the second row are tracking results from frames 21, 31, and 41. The color scheme is the same as above.

main, our method would likely benefit from the addition of contour information to the objective function. It would be relatively simple to render a wireframe of the object and add an additional conditional probability term to the objective function, measuring distance between the projected model contours and detected image contours, similar to that used in [9]. A robust estimator of some sort would be necessary, though, to avoid the function being dominated by spurious edge detections when they can not be detected reliably, or to de-emphasize the contribution of the edge term before gross alignment has been achieved.

Finally, the need for a small number of hand-segmented images is still a limitation of the current methods. However, we have performed preliminary experiments, not described here, using Expectation-Maximization methods to learn appearance models without hand training. The results are quite positive, but are difficult to generalize in our current implementation. Although our current model deals well with many of the domain challenges, effects like extreme changes in lighting or the covering of the target with blood or other fluids would require a self-updating model for tracking over long sequences in practical applications. More generally, non-parametric foreground models and related methods for model updating such as those described in [24] are clearly needed in order to move to objects with changing appearance.

VI. Acknowledgments

We thank the National Science Foundation (NSF), for their support of this work through grants IIS-0205318 and R21 EY016359, and our reviewers, for their comments. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] S. Avidan. Ensemble tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, Feb. 2007.

[2] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Proc. Computer Vision and Pattern Recognition*, pages 568–574, 1997.

[3] M. Cascia, S. Sclaroff, and V. Athitsos. Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models, 1999.

[4] Y. Cheng. Mean shift, mode seeking, and clustering. *Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, Aug. 1995.

[5] N. Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide v 1.1*, November 2007. Available at http://developer.nvidia.com/object/cuda.html.

[6] J.-B. de la Riviere and P. Guitton. Image-based analysis for model-based tracking. In *Mirage*, March 2005.

[7] G. Dewaele, F. Devernay, and R. Horaud. Hand motion from 3d point trajectories and a smooth surface model. In *Computer Vision - ECCV 2004*, volume Volume 3021/2004, pages 495–507. Springer Berlin / Heidelberg, 2004.

[8] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[9] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):932–946, 2002.

[10] D. M. Gavrila. The visual analysis of human movement: a survey. *Comput. Vis. Image Underst.*, 73(1):82–98, 1999.

[11] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 73–80, San Francisco, CA, USA, June 1996.

[12] H. Grabner, P. M. Roth, and H. Bischof. On-line boosting and vision. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 260–267, June 2006.

[13] H. Grabner, P. M. Roth, and H. Bischof. Learning features for tracking. In *Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[14] R. Haralick and L. Shapiro. *Computer and Robot Vision*. Addison-Wesley Publishing, first edition, 1993.

[15] C. Harris. *Tracking with Rigid Objects*. MIT Press, 1992.

[16] I. Intuitive Surgical. http://www.intuitivesurgical.com/index.aspx.

[17] J. Kessenich. *The OpenGL Shading Language v 1.20*, September 2006. Available at http://www.opengl.org/documentation/glsl/.

[18] L. Kukolich and R. Lippman. *LNKnet User's Guide*, February 2004. Available at http://www.ll.mit.edu/IST/lnknet/usersguide.pdf.

[19] Le Lu and Gregory D. Hager and Laurent Younes. A Three Tiered Approach for Articulated Object Action Modeling and Recognition. *Advances in Neural Information Processing Systems*, 17:841–848, July 2005.

[20] S. U. Lee, S. Y. Chung, and R. H. Park. Performance study of several global thresholding techniques for segmentation. *Comput. Vision Graph. Image Process.*, 52(2):171–190, 1990.

[21] H. Li, P. Roivainen, and R. Forcheimer. 3-d motion estimation in model-based facial image coding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(6):545–555, 1993.

[22] D. G. Lowe. Fitting parameterized 3-d models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:441–450, 1991.

[23] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[24] L. Lu and G. D. Hager. A nonparametric treatment for location/segmentation based visual tracking. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[25] S. J. McKenna, Y. Raja, and S. Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17(3-4):225–231, 1999.

[26] J. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7:308–313, 1964.

[27] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 29:65–81, January 2007.

[28] J. M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: An application to human hand tracking. In *In European Conference on Computer Vision*, pages 35–46. Springer-Verlag, 1994.

[29] B. Rosenhahn, C. Perwass, and G. Sommer. Pose Estimation of 3D Free-Form Contours. *International Journal of Computer Vision*, 62(3):267–289, 2005.

[30] A. Ruf, M. Tonko, R. Horaud, and H. H. Nagel. Visual tracking of an end-effector by adaptive kinematic prediction. In *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 2, pages 893–899, Grenoble, France, Sept. 1997.

[31] C. Sminchisescu, A. Kanaujia, and D. N. Metaxas. Bm3e: Discriminative density propagation for visual tracking. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, 29:2030–2044, November 2007.

[32] S. Speidel, G. Sudra, S. Schalck, B. Muller-Stich, C. Gutt, and R. Dillmann. Automatic image-based analysis for context-aware assistance in minmally invasive surgery. In *SURGETICA 2007: Computer-aided medical interventions: tools and applications*, pages 53–62, 2007.

[33] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, Fort Collins, CO, USA, 1999.

[34] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. Camera calibration toolbox for matlab. Published via web. Available at http://www.vision.caltech.edu/bouguetj/calib_doc/.

[35] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, 2004.

[36] L. Xu and M. Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8:129–151, 1996.

[37] T. Yu, C. Zhang, M. Cohen, Y. Rui, and Y. Wu. Monocular video foreground/background segmentation by tracking spatial-color gaussian mixture models. In *Motion and Video Computing, 2007. WMVC '07. IEEE Workshop on*, pages 5–5, Feb. 2007.

[38] C. Zhang and Y. Rui. Robust visual tracking via pixel classification and integration. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 37–42, 2006.