Unified Detection and Tracking in Retinal Microsurgery

Raphael Sznitman, Anasuya Basu, Rogerio Richa, Jim Handa, Peter Gehlbach, Russell H. Taylor, Bruno Jedynak, and Gregory D. Hager

Johns Hopkins Univesity
3400 N. Charles Str., Baltimore, USA
{sznitman,abasu8,richa,rht,bruno.jedynak,hager}@jhu.edu,
{jthanda,pgelbac}@jhmi.edu

Abstract. Traditionally, tool tracking involves two subtasks: (i) detecting the tool in the initial image in which it appears, and (ii) predicting and refining the configuration of the detected tool in subsequent images. With retinal microsurgery in mind, we propose a unified tool detection and tracking framework, removing the need for two separate systems. The basis of our approach is to treat both detection and tracking as a sequential entropy minimization problem, where the goal is to determine the parameters describing a surgical tool in each frame. The resulting framework is capable of both detecting and tracking in situations where the tool enters and leaves the field of view regularly. We demonstrate the benefits of this method in the context of retinal tool tracking. Through extensive experimentation on a phantom eye, we show that this method provides efficient and robust tool tracking and detection.

1 Introduction

Surgical tool tracking has recently established itself as part of the computer assisted intervention community [1,2]. Ophthalmic microsurgery is an emerging research field, and the ability to detect and track retinal tools is very important for automatic vein cannulation, retinal modeling, precise retinal image mosaicking and other microretinal surgery applications. Informally, the objective of visual tracking is to provide an accurate estimate of the parameters or configuration of a tool across time. A general solution involves two subtasks: (i) detecting the tool in the initial image in which it appears, and (ii) predicting and refining (*i.e.* tracking) the configuration of the detected tool in subsequent images [3].

Indeed, few accurate tool detection and tracking systems exists in the context of retinal microsurgery. In [4] tracking is performed frame by frame by using color appearance and a 3D tool model, rendering tracking relatively slow. Similarly, [5] casts the retinal tool tracking as a linear programing problem, achieving good accuracy but also suffering from high computational cost. Gradient-based color tracking in retinal image sequences was also proposed in [2,6], but is highly depend on good initialization. More generally within the context of computer

G. Fichtinger, A. Martel, and T. Peters (Eds.): MICCAI 2011, Part I, LNCS 6891, pp. 1–8, 2011.

vision, combining detection and tracking remains a difficult problem when an object enters and leaves the field of view frequently.

We propose an algorithmic framework that combines tool detection and tracking in an information theoretic setting. Instead of treating detection and tracking as separate tasks, we cast both as a single sequential entropy minimization problem, treating tool detection and sequential localization estimation as one. Consequently, the system presented here consists of a single process, as opposed to two, and hence reduces the need for careful initializations and parameter tuning. Our solution solves detection and tracking as a density estimation problem by using an *Active Testing* (AT) [7,8] optimization strategy. In practice, we demonstrate that our approach provides a feasible and automatic solution to tool detection and tracking, without the need of accurate motion models. While we have chosen to show this method in the context of retinal tool tracking, we believe it is general enough for kinematic tools in a number of other surgical settings.

The remainder of the paper is organized as follows: in Sec. 2 we first describe tracking as a Bayesian sequential estimation problem and how our approach embodies this structure. We then describe how the AT paradigm is used to locate tools in Sec. 3. In Sec. 4, we perform extensive experimentation to validate our approach. Finally, we conclude with some closing remarks in Sec. 5.

2 Tool Tracking

In what follows, we parametrize the surgical tool by (1) the location on the boundary of the image where the tool enters, (2) the angle the tool makes with the image boundary and (3) the length of tool (see Fig. 1(a)). Note that this choice simply reflects known constraints for this application. More formally, the tool configuration space is defined as $Y = (y_1, y_2, y_3) \in \mathcal{S}_1$ where, $\mathcal{S}_1 = [0, P] \times [-\pi/2, \pi/2] \times [\delta, D]$, where P is the length of the perimeter of the image in pixels, δ is a minimal length the tool must protrude before being considered visible and D is the diagonal image length. To model the tool leaving the field of view, we allow a special token $\mathcal{S}_0$ indicating that the tool is not visible (*i.e.* $Y \notin \mathcal{S}_1$). Thus, $Y \in \mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1$.

We cast the tracking problem in a Bayesian sequential estimation fashion. That is, at time t, we consider a random variable Y^t that must be inferred given the image sequence observed up to that time instance, $\mathcal{I}^t = (I^0, \ldots, I^t)$. Using a standard Hidden Markov Model assumption, we formulate this as a Bayesian filtering problem [9]. That is, we specify a prior distribution on Y, denoted $P(Y^0)$, a tool dynamics model, $P(Y^t|Y^{t-1})$, and will sequentially estimate the new tool parameters given the observed images $\mathcal{I}^t$.

Active Testing Filtering: In order to compute the posterior distribution of Y given the history of observations, $P(Y^t|\mathcal{I}^t)$, we make use of the AT strategy. AT is a stochastic optimization technique used for parameter estimation. When trying to determine a set of parameters, $Y = (y_1, \ldots, y_n)$, this technique requires a prior

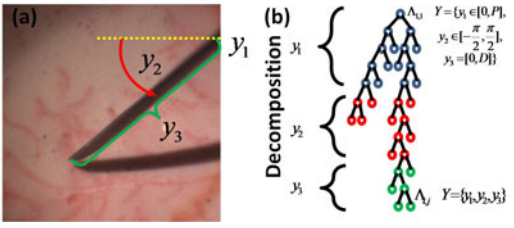

Fig. 1. Parametrization of retinal tool: (a) Visual representation of the tool parametrization (b) Decomposition of tool pose space

probability distribution on the parameters, which in our case will be $P(Y^t|\mathcal{I}^{t-1})$, a set of questions, $\mathcal{X}$, pertaining to the parameters and a parametrization of the parameter space. The main idea is to use the available questions to make the distribution of Y evolve such that it is peaked on the correct parameters, (*i.e.* tool location and pose). AT automatically selects which questions to ask by using an entropic loss function. Algorithmically, this consists in: (i) select a question (*i.e.* an image functional) and subset of the parameter space, (ii) apply this question to the subset selected, (iii) update the probability distribution of the parameters, (iv) select the next question and subset pair that maximizes the mutual information gain, (v) repeat from (ii) until the entropy of the distribution is low. For a more thorough review of AT, see [7,8].

Given this, we propose a general Active Testing Filter (ATF) (see Alg. 1). The user initially provides, some dynamics model, $P(Y^t|Y^{t-1})$ and an initial prior $P(Y^0)$. Then, for an image in the sequence, first compute $P(Y^t|\mathcal{I}^{t-1})$ (line: 3) by using the provided dynamics model. Then treat $P(Y^t|\mathcal{I}^{t-1})$ as an initial prior for the AT localizer (line: 4). This process is then repeated for all images in the sequence

In the following section, we specify the required question set and space parametrization.

Algorithm 1. Active Testing Filtering ($\mathcal{I} = \{I^1, \ldots, I^T\}$)

1: Initialize: $P(Y^0), P(Y^t|Y^{t-1})$
2: **for all** $t = 1, \ldots, T$ **do**
3: $\quad P(Y^t|\mathcal{I}^{t-1}) = \int P(Y^t|Y^{t-1})P(Y^{t-1}|\mathcal{I}^{t-1})dY^{t-1}$
4: $\quad P(Y^t|\mathcal{I}^t) = ActiveTesting(I^t, P(Y^t|\mathcal{I}^{t-1}))$
5: **end for**

3 Active Testing for Tool Localization

To use the AT framework, one must provide a question set and specify a partitioning of the search space. We now describe these specifications.

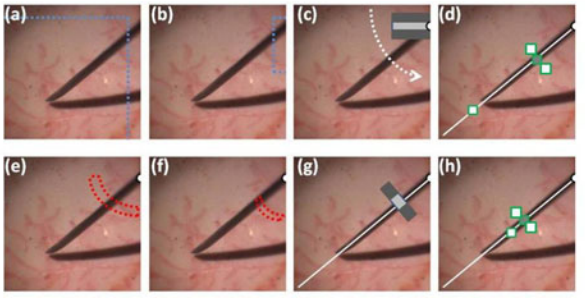

Fig. 2. Question Set. Example images with each image designating the queried region. See text for details on these queries.

Space Representation: Let Λ denote a binary decomposition of the space $\mathcal{S}_1$. That is, Λ is a tree of sets, $\Lambda = \{\Lambda_{i,j}, i = 0, \ldots, M, j = 0, \ldots, 2^i - 1\}$ (see Fig. 1(b)). The root of tree is $\Lambda_{0,0} = \mathcal{S}_1$. The decomposition of the tree is performed by splitting one coordinate of Y at a time, until a desired resolution, at which point we repeat the procedure for another coordinate (*e.g.* split y_1, then y_2 and so on). That is, the partition forms conditional subtrees: the first subtree decomposes the first parameter, while the second subtree decomposes the second parameter conditioned on the first, and the last subtree partitions the final parameter conditioned on the two first parameters. The subtrees are color coded in Fig. 1(b). It is easy to show that any level of the tree forms a covering on $\mathcal{S}_1$, *i.e.* , $\mathcal{S}_1 = \Lambda_{0,0} = \bigcup_{j=0}^{2^i-1} \Lambda_{i,j}$.

Query Set: To determine the target configuration, a set of questions or tests, $\mathcal{X}$, pertaining to the target must be provided. This consists in associating each node $\Lambda_{i,j}$ with a set of pose-indexed queries that provide information on whether or not the target has configuration contained within the node at hand. For each node $\Lambda_{i,j}$, we define a set of questions, $X_{i,j} = \{X_{i,j}^1, \ldots, X_{i,j}^K\}$ where $X_{i,j}^k$ is some image functional; $X : I \mapsto \mathbb{R}$. Finally, all questions are assumed to have homogeneous responses such that,

$$P(X_{i,j}^k = x | Y = y) = \begin{cases} f_o^k(x) & \text{if } y \in \Lambda_{i,j} \\ f_b^k(x) & \text{if } y \notin \Lambda_{i,j} \end{cases} \tag{1}$$

where f_o^k and f_b^k are two distributions of responses, corresponding to the case where the tool configuration is in the space queried, and when it is not. Since the response, x to the query $X_{i,j}^k$ will be in $\mathbb{R}$, both f_o^k and f_b^k will be modeled as Gaussians. The parameters of these distributions (*i.e.* μ, σ^2) are learned from separate training images representative of test sequences.

We now specify the query set, $\mathcal{X}$. In total, we form six queries. That is for each node in Λ, we have $X_{i,j} = \{X_{i,j}^1, \ldots, X_{i,j}^6\}$. We let $X_{i,j}^1$ count the proportion of pixels deemed tool like, by using a color model learned from data, over a region determined by the interval of y_1 in $\Lambda_{i,j}$. An example is shown in Fig. 2(a)-(b),

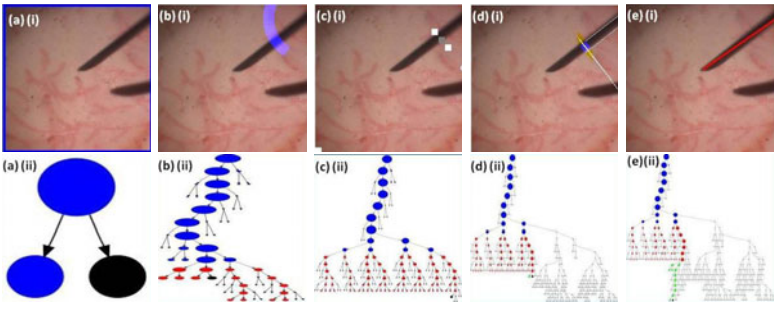

Fig. 3. Active Testing Iterations. Each image pair (top and bottom row) show a query being evaluated and the corresponding state of the AT tree at that point in time, respectively. See text for details.

where each image shows a different $\Lambda_{i,j}$ being queried by $X^1_{i,j}$. $X^2_{i,j}$ also queries a part of the y_1 interval by applying a small template to the image boundary centered on y_1, and compute the minimum template score when varying the template orientation (see Fig 2(c)).

In order to estimate y_2, we form $X^3_{i,j}$ and $X^4_{i,j}$. Given a fixed tool boundary point, $X^3_{i,j}$, computes the proportion of tool pixels (as in $X^1_{i,j}$) which are present in an arc defined by the pose interval of y_2 in $\Lambda_{i,j}$ (see Fig. 2(e)-(f) for two examples). $X^4_{i,j}$ again applies a template match perpendicular to the average angle in the interval of y_2 in $\Lambda_{i,j}$ (see Fig. 2(g)).

Estimating y_3 is achieved by queries $X^5_{i,j}$ and $X^6_{i,j}$. At this point, both the tool boundary point and the angle are assumed to be fixed. Estimating if the length of the tool is in the interval of y_3 in $\Lambda_{i,j}$ is done by computing the difference in average pixel intensities at the upper bound and lower bounds of y_3 in $\Lambda_{i,j}$ (see Fig. 2(d)-(h)). Finally, $X^6_{i,j}$ performs a template match according to the parameters specified in $\Lambda_{i,j}$.

4 Experiments

We choose the prior of Y to be uninformative and let $P(Y^0) = P(\{y_1 \in [0, P], y_2 \in [-\pi/2, \pi/2], y_3 \in [0, P]\}) = 1/2$, indicating equal likelihood that tool is or is not in the image. Here, we use a simple linear dynamic model of the form, $Y^{t+1} = AY^t + \mathcal{N}(0, \alpha)$, where A is the dynamics transition matrix. In the experiments that follow, A is augmented to allow velocity estimates to be compounded in the new prior. Given that we know that the tool will enter and leave the field of view often, we expect the dynamics model to be violated regularly. While this may induce inappropriate priors $P(Y^t|\mathcal{I}^{t-1})$ at each step, the AT framework will correct for this automatically. Note that, the AT model is trained on a separate and representative training set of 50 images, where the tool pose has been annotated. Also, since we are only interested in tracking the tool when it is in focus and the depth of field is particularly small on retinal

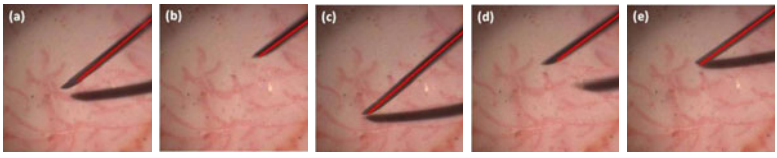

Fig. 4. Snapshots of detection and tracking during an image sequence

microscopes, the tool scale exhibits relatively little variability. Hence, we will assume very limited tool scale changes.

Experimental Setup: Similarly to [4,2], we recorded two video sequences of a retinal tool interacting with a phantom retinal membrane. The sequence consists of 400 frames of size 256×256. In these sequences, a retinal tool (needle) is initially present in the field of view and moves regularly. The tool leaves and re-enters the field of view multiple times throughout these sequences. In images that contain the tool, the locations have been manually annotated for quantitative analysis evaluation. All algorithms were tested on a standard 2.3GHz PC.

To illustrate the AT step in our ATF approach, Fig. 3 shows both the queries asked and the evolution of Λ at selected iterations during this process. The top row shows which query is being evaluated with the queried region highlighted in each image. The bottom row shows the state of Λ at that point. The area of each node is proportional to the mass contained in that pose subset, while the color of each node represents which coordinate is being refined (as in Fig. 1).

Initially, only the root $\Lambda_{0,0}$ exists and a query is evaluated on the entire pose space. Having created children (Fig. 3(a)i-ii), the size of Λ consists of three nodes. After a few queries, the tree has grown and refined itself past the first coordinate y_1, onto y_2 and y_3 (Fig. 3(b)(c)i-ii). Eventually the correct y_2 parameter (Fig. 3(d)i-ii) is located, leading to a valid tool detection (Fig. 3(e)i-ii).

Having detected the tool parameters, we use the dynamics model to compute a new prior that seeds another round of the AT step. As such, no separate mechanism is required to initialize tracking, or filter responses. Fig. 4 shows some selected images of the tool being tracked. When the tool moves beyond the domain of the image, tracking is abandoned, correctly detecting that no tool is present. The object then reappears in the field of view and is automatically reacquired. Notice that when the tool tip and tool shadow approach together, the tool object is still correctly tracked (Fig. 4(c)(e)).

Comparison: To evaluate the performance of the ATF approach, we compared it with two methods: Simple Detect (SD) and Simple Detection and Tracking (SDT). Both these methods are specially tailored using *a priori* knowledge of the appearance of the tool and background in the experimental image sequences.

In SD, all pixel positions are tested to detect the tip of a long shaft. The pixel intensity threshold for separating tool and background was tuned empirically so that no false detections occur. Once it is detected, the tool is tracked using the tool position from the previous image for initialization.

Method	Accuracy Error				Detection			Time (ms)
	y_1	y_2	y_3	Tip	TPR	FPR	Precision	
ATF	**3.51 (0.13)**	**1.79 (0.1)**	**12.67 (0.85)**	**11.51 (0.84)**	**0.839**	6.4×10^{-6}	**0.661**	7.21
AT	5.53 (0.70)	2.41 (0.13)	14.16 (1.02)	13.49 (0.81)	0.811	6.1×10^{-6}	0.615	25.21
SD	84.87 (2.45)	29.47 (0.81)	20.33 (0.94)	15.19 (0.79)	0.778	7.1×10^{-6}	0.547	26.67
SDT	52.58 (7.41)	11.09 (0.92)	21.24 (2.3)	11.64 (0.24)	0.834	$\mathbf{7.4\times10^{-6}}$	0.596	4.8

Fig. 5. Performance Results

In a similar spirit to [2], SDT includes a tracking loop after detection (as performed by SD). The tool tracking loop consists of the following steps. First, for a new image, the displacement of the tool is found by searching along a line perpendicular to the tool. Second, the new tool angle is measured. The maximal angle variation is set to 35 degrees for limiting the computational search cost. Finally, the tool tip is found by searching along the tool shaft for the intersection of the tool and the background (the intersection is determined by thresholding the intensity level using empirically chosen values)[1].

In our comparison we observe three aspects of performance. First, we compute the error in the estimates of each parameter and the tool tip position. This error is computed by using the annotated ground truth. Similarly, we compute statistics which are typically observed in detection and localization tasks. We consider a correct detection to be one which estimates the tool tip location to within 10 pixels of the ground truth. We then compute the *true positive rate* (TPR), the *false positive rate* (FPR) and the precision for each approach. Finally, we report the computational time of each algorithm. Given these performance measures, we compare the following four algorithms: the Active Testing algorithm (AT), both comparison methods described above, SD and SDT, and the ATF algorithm.

Fig. 5 summarizes our experimental results. For the accuracy error, we report the means and standard errors for each tool parameter and tool tip. Notice that in general all the methods proposed provide more or less the same detection accuracy. Naturally, we see that detection is significantly slower than tracking with both AT and SD running much slower than tracking methods, confirming the advantages of tracking strategies over tracking by pure detection approaches.

From the algorithms tested, ATF appears to outperform other methods and yet remains computationally efficient (*i.e.* over 90 fps). In the domain of accuracy, the ATF approach estimates the tool parameters more accurately and consistently. When compared to SDT, ATF is generally capable of determining the configuration of the tool more accurately. This improvement can be attributed to the sequential parameter estimation that the active testing framework conducts. By estimating the first parameter, then the second and so on, each parameter is individually estimated accurately. This is in sharp contrast to the more direct SDT approach which locates the tool tip, and then estimates the necessary parameters.

[1] See project website for additional information.

5 Conclusion

We have proposed a novel approach to tool tracking and detection. By treating both problems in a sequential tool parameter estimation setting, our framework minimizes the entropy of the joint distribution over the tool parameters and the available questions provided. A full detection and tracking algorithm has been outlined, and we have empirically shown that the proposed method is capable of detecting and tracking retinal tools efficiently and robustly in cases where the tool enters and leaves the field of view frequently. As part of our future work, we are looking at extending this framework to tracking multiple targets such that this approach may be applicable in a larger number of settings.

Acknowledgments. Funding for this research was provided in part by NIH Grant R01 EB 007969-01, an unrestricted grant by RPB (Wilmer Eye Institute), internal JHU funds and a research subcontract from Equinox Corporation.

References

1. Voros, S., Long, J.A., Cinquin, P.: Automatic detection of instruments in laparoscopic images: A first step towards high-level command of robotic endoscopic holders. IJRR 26, 1173–1190 (2007)
2. Dewan, M., Marayong, P., Okamura, A.M., Hager, G.D.: Vision-based assistance for ophthalmic micro-surgery. In: Barillot, C., Haynor, D.R., Hellier, P. (eds.) MICCAI 2004. LNCS, vol. 3217, pp. 49–57. Springer, Heidelberg (2004)
3. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Computing Surveys 38(4) (2006)
4. Sznitman, R., Rother, D., Handa, J., Gehlbach, P., Hager, G.D., Taylor, R.: Adaptive multispectral illumination for retinal microsurgery. In: Jiang, T., Navab, N., Pluim, J.P.W., Viergever, M.A. (eds.) MICCAI 2010. LNCS, vol. 6363, pp. 465–472. Springer, Heidelberg (2010)
5. Pezzementi, Z., Voros, S., Hager, G.D.: Articulated object tracking by rendering consistent appearance parts. In: IEEE international Conference on Robotics and Automation, pp. 1225–1232. IEEE Press, Piscataway (2009)
6. Burschka, D., Corso, J.J., Dewan, M., Lau, W., Lia, M., Lin, H., Marayong, P., Ramey, N., Hager, G.D., Hoffman, B., Larkin, D., Hasser, C.: Navigating inner space: 3-D assistance for minimally invasive surgery. Robotics and Autonomous Systems 52(1), 5–26 (2005)
7. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. IEEE TPAMI 18(1), 1–14 (1996)
8. Sznitman, R., Jedynak, B.: Active testing for face detection and localization. IEEE TPAMI 32(10), 1914–1920 (2010)
9. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, Cambridge (2005)