An Online Learning Approach To In-Vivo Tracking Using Synergistic Features

Austin Reiter and Peter K. Allen

***Abstract*— In this paper we present an online algorithm for robustly tracking surgical tools in dynamic environments that can assist a surgeon during in-vivo robotic surgery procedures. The next generation of in-vivo robotic surgical devices includes integrated imaging and effector platforms that need to be controlled through real-time visual feedback. Our tracking algorithm learns the appearance of the tool online to account for appearance and perspective changes. In addition, the tracker uses multiple features working together to model the object and discover new areas of the tool as it moves quickly, exits and re-enters the scene, or becomes occluded and requires recovery. The algorithm can persist through changes in lighting and pose by using a memory database, which is built online, using a series of features working together to exploit different aspects of the object being tracked. We present results using real in-vivo imaging data from a human partial nephrectomy.**

I. Introduction

Minimally invasive surgery (MIS) has become a standard operating choice for surgeons in recent years due to its low risks and high rewards. Patients and doctors alike prefer fewer scars and smaller incisions with quicker healing times. Currently, MIS requires multiple incisions and multiple people in the operating room, all trying to work together, but with much effort.

While most laparoscopic procedures require multiple incisions, two important new and emerging surgical paradigms are Single Port Access surgery (SPA) and Natural Orifice Translumenal Endoscopic Surgery (NOTES) [1]. In these methods, only a single incision or a natural body orifice is used to introduce surgical imaging and tooling devices into the body. To support these new methods, a whole new class of robotically controlled surgical instrumentation [2] is required. We have recently designed and fabricated such an in-vivo robotic surgery platform we call the IREP: Insertable Robotic Effector Platform for SPA surgery. The IREP (see figure 1) is a 15mm diameter package that contains stereo cameras and 2 snake-like robotic arms that can be deployed inside the body to perform surgical procedures. Details of the IREP can be found in [3].

The requirements of using a device such as the IREP put a heavy emphasis on human interface issues: how does the surgeon control the platform which typically has multiple degrees of freedom and the need to image wide field of view surgical sites with varying resolution and lighting? For example, the IREP has two 7-DOF robot snake arms and a 3-DOF stereo imaging system (pan, tilt, elevation) which needs to be controlled by the surgeon. This paper describes our initial efforts in building computer vision tools that can assist the surgeon during an IREP procedure. As the surgeon is mostly focused on using the tooling and effectors, our goal is to provide salient 2-D and 3-D information from the vision module to the surgeon in real-time. We want our imaging operations to be performed autonomously to free the surgeon from directing and moving the camera.

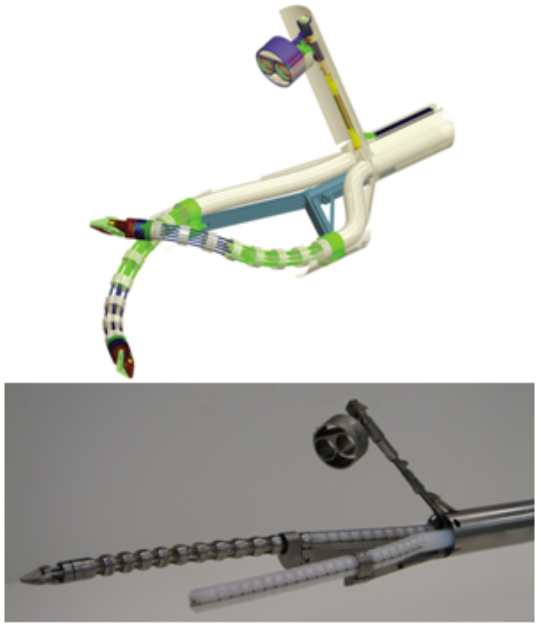

Fig. 1. (Top) IREP robot design in working configuration. (Bottom) IREP Robot stereo vision module with actuation and one 7-DOF snake arm extended.

A common problem in the operating room (OR) during an MIS procedure is the interaction between the surgeon and the camera operator, as it is often difficult to anticipate the motions of the surgeon. Furthermore, time is often a factor as the surgeon must work quickly while blood vessels and arteries are temporarily clamped to accomplish a procedure, and the surgeon and camera operator don't have time to move slowly. Therefore, in order to make these robotic devices equipped for the OR, there must be an automated method of controlling the camera to follow the surgeon's movements in real-time to alleviate this issue. The tracking algorithm presented in this paper is a step forward in providing a robust tracker that can persist in a dynamically-changing environment with quick movements and frequent tool recoveries in order to

A. Reiter and P. Allen are with Dept. of Computer Science, Columbia University, 450 Computer Science Building, 1214 Amsterdam Avenue, New York, NY 10027-7003 areiter, allen@cs.columbia.edu

keep the camera centered where the surgeon desires. Linking this tracker with the camera control system of the IREP will allow the surgical robot to perform automated visual servoing of surgical tools.

The conditions of tracking in-vivo during minimally invasive surgery are quite different from other domains where tracking algorithms are actively being developed, such as in the mobile robot or surveillance domains. The challenges provided by working in a tight space inside the body are numerous and difficult to overcome. Changing lighting and severe pose changes are common. Because the camera is situated very close to the workspace, small 3D movements translate to large 2D pixel displacements in the image frames, and these typically occur quite quickly. Also, accounting for camera motion is challenging due to the deformable surfaces of organs that are frequently moving. The movements of the surgeon that we wish to track are often very erratic, and due to the limited field-of-view, exiting and re-entering the scene must be accounted for, which breaks continuous tracking algorithms that don't learn and can't recover.

Section II reviews some previous work in the area of surgical tool tracking. Section IV describes our algorithm in detail as well as reasons for the choices we made in designing the different modules. In section V, we present experimental results from two human partial nephrectomy video sequences to show how the tracker performs in a real environment. Section VI ends with conclusions and future areas of research to build onto this tracking framework.

II. Previous Work

There has been much progress in the field of tracking surgical instruments for use in automated visual servoing during laparoscopic surgery. In the medical vision field, typically either color or texture is used, and in cases where information about the tool is known *a priori*, a shape model can be used to confine the search space [4] [5] [6].

A common method is to design a custom marker, as in [7] [8] [9], to assist in tool tracking. Here, the authors make the argument that shape or geometry are not reliable enough for tracking, and so they design a color marker by studying the HSV color space to determine what color components aren't prevalent in typical surgery imagery, and then fabricate their own marker to be placed on the tool. A training step creates a kernel classifier which can then label pixels in the frame as either foreground (tool) or background. Similarly, the authors in [10] design a marker with 3 stripes that traverse the known diameter of the tool. This allows the estimation of depth information as well.

Color may be exploited without custom markers, as in [11], in which the authors utilize different color signatures of organs and instruments to classify individual pixels by training on a large sample of pixels from endoscopic sequences. A Bayesian classifier maximizes the *a posteriori* probability of the class assignment in order to distinguish organ from instrument. Sometimes, simple assumptions can be made about the scene, such as determining "grey" regions and labeling them as the instrumentation [12] [13] [4]. The authors contribute a new definition of color purity component and aim to extract boundaries of nearly uniformly grey regions to develop the idea that the color saturation is the most discriminate attribute for grey region segmentation, and in so doing, define a new definition of saturation.

Another technique to aid in tracking is to affix assistive devices to the imaging instrument itself. In [14], a laser-pointing instrument holder is used to project laser spots into the laparoscopic imaging frames. This is useful for when the tools move out of the field-of-view of the camera. The laser pattern projected onto the organ surface provides information about the relative orientation of the instrument with respect to the organ. Optical markers are used on the tip of the surgical instruments, and these markers used in conjunction with the image of the projected laser pattern allow for measurements of the pointed organ and the instrument.

Prior information of the surgical tools may be used to confine the search space for the instrument [5]. Here, the authors perform a calibration step to define the 3-D insertion point of the instrument into the abdominal cavity. This gives shape considerations to confine the search space for the instrument and helps achieve real-time processing.

Offline learning has been used to combine multiple features together into a strong feature framework [6], wherein the authors extract color and texture features and train offline. The object configuration is estimated by using a prior geometric model of the object and maximizing the correlation between the rendering and the probability map. Online learning has been used in [15] for feature tracking which learns the representation for corner features and adapts the feature representation for soft tissue tracking. The tracking uses the classification technique in [16] to find optimal solutions and then estimates dynamic tissue deformation.

III. Overview

Our work seeks to combine these two ideas of using multiple features with an online approach to both disciminate the object from the scene as well as account for changes in the object's appearance over time. Unlike the work in [6], we adjust feature representations online which allow us to discover new parts of the object as it moves and turns in a dynamically-changing environment. The basis of our algorithm lies in the following 3 assumptions:

1) One feature alone will not suffice for long-term, robust tracking
2) Features *working together* (**synergy**) are better than features *working alone*. In this sense, features will **flow** into one another (rather than work independently and force their combination in the final step). We call this idea **Feature Flow**.
3) Learning feature representations online is important for long-term, robust tracking

We keep a database of track states over time, storing gradient-based features (i.e., corners, edges, etc.) of the object being tracked. We use these feature locations to estimate the position of the object in the current frame (similar to [17]), and then compute a likelihood map using a

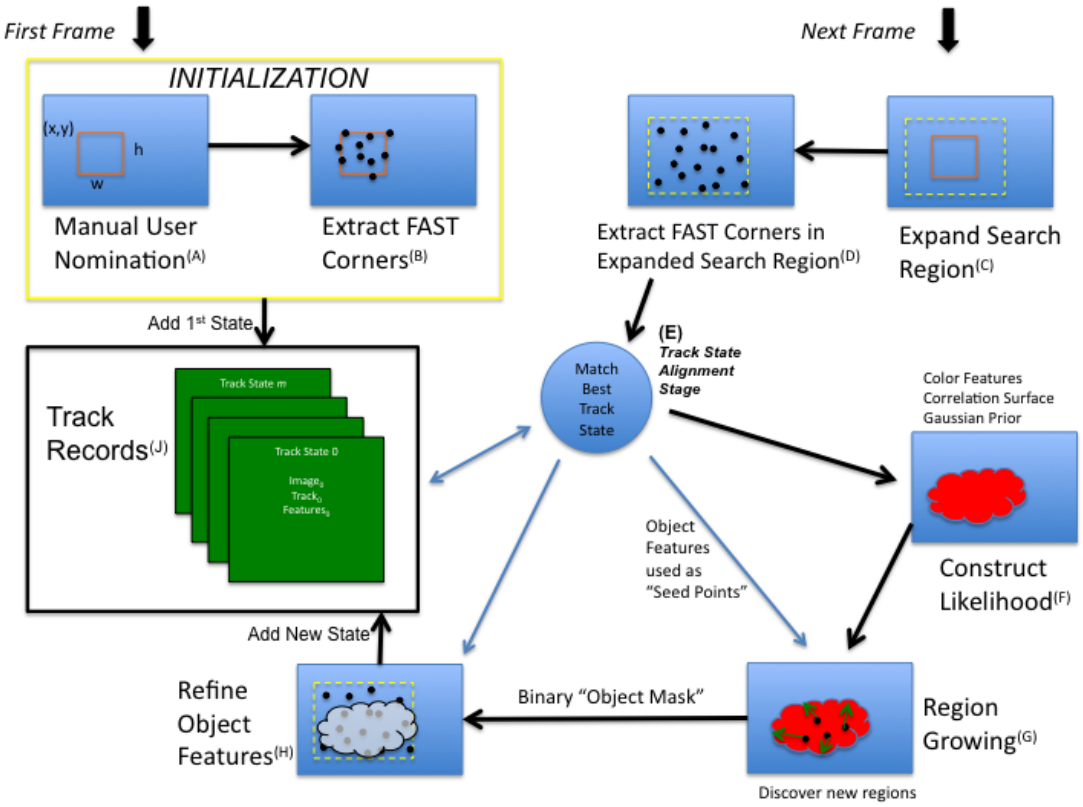

Fig. 2. Feature-Flow tracker diagram showing the individual components of the tracker and how they feed into each other. Each module/box is labeled with a letter, which is referenced in the text to more easily visualize each step of the procedure. Small-textured features (corners) are used to estimate the track position using a database of previous track states. New regions are then discovered using features such as color and large-scale texture (correlation) in a combined, weighted-likelihood probability map. Inliers from the alignment step are used in a region-growing procedure inside the combined likelihood to identify new pixels of the object not seen before.

combination of other synergistic features (i.e., color, texture). The **likelihood map**, defined as an image specifying the probability of a pixel being part of the tracked object, assists the tracker in identifying new regions of the object. We use the feature locations as "seed points" in a region-growing algorithm, within the likelihood map to *flow* into new parts of the object, and then store this new information in our database.

In this way, gradient-based features become the main cue to the new track location, and color and texture features are used to refine and expand the analysis. Previous work [18] suggests that this flow of processing is more akin to human visual processing, where intensity changes (i.e., edges, corners) are the raw primitives extracted at the earliest stages of vision, and are then used as building blocks to group together using other visual clues to form more useful visual representations. If we detect a region of the image that exhibits similar color properties to the object we are tracking, but no mathematical transformation can be made to warp this region to our object, then it is likely that this region is not our object. On the other hand, if a region of the image can be matched to our object-of-interest, but the color properties have changed, it is much more likely (than the previous case) that it is the same object, but under different viewing conditions. Therefore, this order of processing was chosen carefully.

In the following two sections, we will describe our method for achieving each of these ideas and layout the framework for our tracking algorithm. While we have chosen a set of features that are tailored to our task, we note that one of the strengths of our system is its flexibility to use other feature sets within the same framework.

IV. Feature Flow

Figure 2 shows the full Feature-Flow tracker in diagram form; each module will be referenced in the text below by its corresponding letter in this diagram, for the reader's reference. We seek to keep a list of **track states** (J) to serve as a memory for the tracking system, which are separated in time, keeping a representative set of samples of the object as it moves, and this is constantly updated as time progresses. A track state consists of: (1) the *image* frame corresponding to the time at which the track state was added to the database; (2) the *region-of-interest* representing the track location and size as it was detected in the image frame; and (3) the set of *features* representing the track state, which we use to match on each new frame. The tracker begins with a manual user nomination (A) consisting of an image patch for the target we wish to track (i.e., a surgical tool). We extract a set of

FAST corners [19] within this track region *only* (B), and add an initial state to our track database. Note that if the object is not sufficiently textured, corner features are not ideal, and in this case, edges may be safely substituted.

The algorithm works as follows: on each frame, we take the previous track region and construct a slightly expanded search region-of-interest (ROI) in the current frame (C), in which we extract features (D). We want to find the best track state match (E) in the database to the current frame, and so for each track state we:

- Perform feature matching using normalized cross-correlation (NCC) of small image patches centered at each corner feature against the features stored in the track state.
- Estimate an alignment between the potential feature matches, eliminating outliers using a variant of RANSAC [20], called M-SAC [21].
- Evaluate the alignment by warping the track state image patch (that part of the image represented by the track stored in the state) using the estimated transform computed above, and computing the NCC as a similarity score. We use an affine model here.
- Take the track state with the highest alignment similarity score as the **best match** to the current frame.

In essence, we could stop here and track frame-to-frame using this technique. This would, and did, perform decently well if the view of the object doesn't change over time. However, as mentioned earlier, a common problem related to in-vivo environments occurs when the tool moves too much, frequently showing different views, and the current method will guarantee you to only be able to lock-on to that part of the object you were originally supplied with in the initialization stage. The result is that we eventually lose the track because the information stored in the database is no longer present in the image frames, and the tracker has nothing else to go on to identify the object. See figure 3 for an example.

To be able to deal with this issue, we chose to combine multiple additional features together into a single likelihood map, representing the probability of each pixel being part of the object, so that we could discover new regions of the object that are similar to what we are currently tracking. The alignment estimation step (done when we are matching to the best track state) yields a set of inlier feature locations that are presumed to be part of the object to be tracked. The inliers are used as *seed points* in the likelihood map in a region-growing algorithm, so that we can **flow** [hence the term Feature-Flow; we flow from one feature (corners) into others (those that form the likelihood)] into new parts of the object. This provides us with an object mask, which we can then use to retrieve the features associated with this image region as a new addition to the track state database.

It's important to note here that the features we retrieve from the object mask will yield new locations that weren't possible to identify with only the feature matching/alignment routines above. They are features that aren't in any of the track states in the database, but are probabilistically-likely to be a part of the object, and so we wish to keep them. This is how we learn the object online.

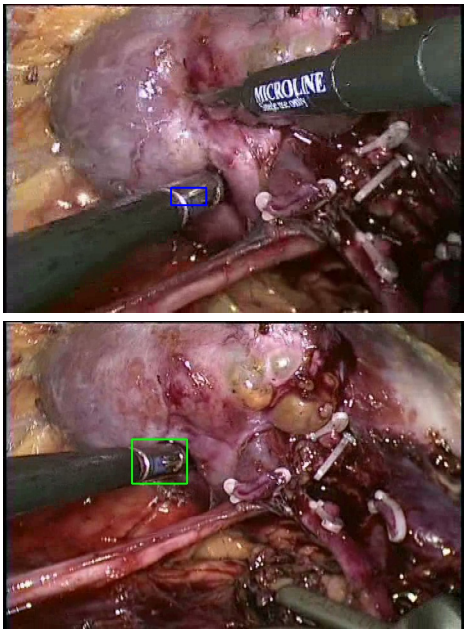

Fig. 3. (Top) Without likelihood discovery, many frames after initialization, the tracker is locked-on to the same part of the object without the ability to discover new regions. As the tool turns significantly, we aren't able to pick up other regions of the same object. (Bottom) With likelihood discovery, we can capture new parts of the object.

Optical flow was considered as the method to estimate the motion parameters of the track ROI. The most widely used formulation [22] assumes a simple translation between successive frames, which does not suffice in the in-vivo environment due to significant changes in scale, rotation, and skew over time of the tool. We don't just want frame-to-frame localization, but rather, we want to be able to match into a database with entries arbitrarily far back in time. Further modifications [23] were made to minimize distortions in flow, which provides for more degrees-of-freedom. However, this is known to be very sensitive to noise. In addition, large erratic displacements tend to cause problems for even the best optic flow algorithms [24], and this issue is guaranteed to be present in surgical in-vivo environments. Because these techniques often require numerical differentation to remain reasonably accurate across time, full-scale warping (as in our method) tends to provide the most accurate localization when fast motions are necessary.

A. Constructing the Likelihood Map

In the likelihood construction stage (F), we wish to define areas of the image which are likely to contain the object we are tracking (foreground) with high values and everything else (background) with low values. The input to this routine is the list of inlier features that come out of the best track state alignment along with the initial estimated track ROI.

We want the likelihood to simultaneously represent different aspects of the object to increase our chances of discriminability. Each of k features will construct its own

individual, probability map, P_k, and the overall likelihood, P_i, is constructed as a weighted-sum of the individual probability maps:

$$P_i = \frac{\sum_k w_k P_k}{\sum_k w_k} \quad (1)$$

where w_k, for each of the k features, represents the corresponding weight on that likelihood feature map. For this paper, k = 3; (1) a Gaussian prior representing our initial estimate of the track location from the corner feature alignment, (2) a large-scale texture analysis, and (3) a color analysis. We have initially weighted each feature equally, but note as future work research into computing dynamic optimal weights based on the filter responses. Also, other features can be used here to compute likelihood maps, with the tradeoff of computational processing; we wish to keep our tracker running in real-time, and so select the minimum number of features possible to achieve this task. It is also possible to speed up this process by utilitizing the parallel-properties of GPUs in computing these features while keeping computational efficiency [6].

1) Gaussian Prior: We use a prior on the likelihood map using the current track estimate. We define a 2D elliptical Gaussian in the region where we believe the track is currently located, from the track state alignment stage (E). The Gaussian is parameterized so that it is centered at this estimated pixel location and is shaped based on the rectangular dimensions of the track match. Therefore, the mean of the Gaussian is the estimated 2D center of the track patch ROI, and the standard deviation in the x-dimension is half of the width of the estimated track patch, and in the y-dimension it is half of the height of the estimated track patch. We also label what we believe to be the background (i.e., everywhere else) with a small, constant probability, but still a non-zero value so it is possible to recover pixels in those areas.

2) Correlation Surface: Next, we look at large-scale texture of the object. Using the best-matched track state and the corresponding affine transform to the current frame, we create a **warped track image** (the part of the image that was stored in the track state, corresponding to the track region only, warped to the current frame) and compute a correlation surface against the current frame. Warping the track ROI is useful because typical NCC surfaces are created with the template as last seen. This could be from much earlier, and even though the small corner features were matchable, the overall structure of the object might have changed markedly. The strength here is that we have an estimate of how that track image might look like in the current frame (i.e., how it has warped), and this produces a much more accurate correlation surface. We convert this into valid probability values by setting all negative correlation scores to zero, and then we update the overall likelihood map.

3) Color Features: Finally, we wish to exploit color features using [16] to compute a set of "tuned" features from a collection of seed features on the current frame. Linear combinations of the RGB color space are used to compute local image window histograms forming the initial seed features. To create the tuned features, we normalize the histograms of each feature descriptor by the number of elements in it, thereby forming a probability distribution function (PDF) on each feature. This is done for all candidate features, in both the foreground and background separately, to achieve class-conditional PDFs. Foreground pixels are collected within the proposed object window and background pixels are collected within a slightly expanded window border around the object box, sampled locally outside the target area. The tuned feature is formed from a log-likelihood ratio of the PDFs:

$$L(i) \quad = \quad \log \frac{\max\{p(i), \delta\}}{\max\{q(i), \delta\}} \quad (2)$$

where p is the object probability, q is the background probability, and δ prevents taking a log of or dividing by zero. This is designed to map object pixels to positive values and background pixels to negative values. Feature discriminability is evaluated using a two-class variance ratio, where variance for the foreground class p is defined as:

$$var(L; p) = E[L^2(i)] - (E[L(i)])^2 \quad (3)$$

and similarly for the background class q. The variance ratio of the log likelihood function is then:

$$VR(L; p, q) = \frac{var(L; (p+q)/2)}{var(L; p) + var(L; q)} \quad (4)$$

and all color features are sorted based on variance ratio and the top one chosen to exploit the current frame. The variance ratio is a scalar value which identifies the discriminability of that color feature, and so sorting on this measure allows us to obtain the most discriminate feature.

We take the color features and corresponding likelihood distribution (i.e., a look-up table of color features to likelihood values) of this top color feature, and create a likelihood image of the entire frame to see how this *best* color feature can rank pixels throughout the whole image. Again, we update the overall likelihood map with the corresponding weight for this feature.

B. Exploiting the Likelihood Map

Now that we have a composite likelihood map made up of several, descriptive features, we use the inliers that came from the best track state match alignment estimation as **seed** points in a region-growing procedure (G) within the likelihood map to eliminate further outliers (if they fall in probabilistically low areas). We also can discover new areas of the object that we haven't seen before. The inliers help **flow** into new areas of the object that we can automatically discover as being part of the same object we are currently tracking. This results in a binary mask of object pixels, which can then reduce the initial set of dense, raw features extracted from the original expanded search ROI (C), and only take

those that fall within the object mask, better signifying "object features" (H). These features are the ones added to the new track state. By adding a grown region of the object, we can then capture new corner features in these new regions and store in our memory. The overall track ROI can then be updated to this new region, to start again on the next frame.

V. Experimental Results

Our tracker was tested using in-vivo imagery from a human partial nephrectomy. We chose a challenging sequence to test the tracker's abilities against changing lighting conditions, pose and perspective changes, rapid movements into and out of the camera's field-of-view, and changing backgrounds due to movements of the organs and bleeding. The initial track was seeded manually by the user providing both a position and size of an image patch to track containing a surgical tool.

We set a maximum number of m (typically 4-5) states to serve as a ring buffer, and we define a constant time step δ ($\sim$1.5-2.5secs) to serve as the update rate for adding new states. We add new states until the memory is full, and then replace the oldest entry. This method works well on a frame-to-frame basis in a real-time setting, with efficient computer memory usage. More complex memories could be used to increase performance, such as a log-distribution in time to keep both a long-term and short-term memory, or extracting something unique about the object (i.e., a 3-D viewpoint) and keeping a model of the object spatially, rather than temporally.

One of the strengths of this system is the ability for it to perform **track recovery**. Using the last known location of the object, we construct a slightly expanded search region ($\sim$1.5 times the previous dimensions) to look for the object. We also take into account previous consecutive failures, and grow the search region dynamically through time so that it gets larger as we miss the object more; this becomes important for tool recovery when the track is lost, is moving fast, or leaves the camera's field-of-view.

Our goal was to keep the tracker running at real-time rates, which we tested on a standard PC running Windows XP with a 2.3 GHz quad-core processor. The tracker runs at approximately 12fps, depending on the size of the object patches in the database as well as the number of states in the database at any given time. However, we found that the tracker was robust enough in finding the object of interest that we could actually skip several frames between updates; therefore, processing every third frame gave us the same qualitative results as processing every frame, effectively tripling the run-time speed to beyond standard video rate of 30Hz.

In figure 4 we show the individual and overall likelihoods (columns 2-4) as well as the tracked frame (column 1, on the left). The colormap for the likelihoods is designed to map high probability values to red colors and low probability values to blue colors. This sequence also required recovery several times as the tool exited the frame and then returned, and each time we were able to recover due to the dynamically-expanding search window.

TABLE I
Performance Comparison of Different Features

Features	Continuous Tracking Time
No Likelihood Discovery	1:01
NCC (No Warp)	1:16
Color (Alone)	1:44
NCC (With Warp)	1:55
Color and NCC (With Warp)	2:17

We performed a study on this video sequence to show the effects of using a synergistic feature likelihood to discover new parts of the object on overall tracking performance, in terms of how long the system was able to keep track. Table I shows the overall tracking time achieved with different choices of features, each starting from the beginning of the video sequence and run until the track is lost or the sequence terminates. The first row is with the corner tracker alone (i.e., no likelihood discovery stage). This is the case mentioned above, in figure 3, and after 1 minute and 1 second, the track was lost completely. The second row is using the NCC as the only likelihood feature, but without warping the track patch to the current frame. Recall our earlier point about the improvement of the correlation surface if the track patch is warped with the estimated track state alignment to the current frame; here the system tracked for 1 minute and 16 seconds, roughly 1/2 the total video sequence time. Next, we show using the color features alone to create the likelihood (third row), where the tracker improved to 1 minute and 44 seconds, but still 33 seconds short of tracking the entire sequence. The fourth row is with the NCC feature using a warped track patch, and we received an improvement to 1 minute and 55 seconds of tracking time, but still not enough to track the entire sequence without losing the object. Finally, the last row shows the full likelihood discovery with both color features and the NCC surface produced from the warped track patch, where we were able to succesfully track the entire sequence of 2 minutes and 17 seconds. Although the warped NCC was able to produce a reasonable result, we still needed the additional color features to track through the full sequence, thereby showing the strength of the synergy of *multiple* features.

The environment was challenging due to many erratic movements, frequent exits from the imaging field-of-view, and occlusions due to moving organs or bleeds. We successfully tracked a sequence for approximately 2 minutes and 17 seconds, to show how the tracker would work in a real enivronment. See `http://www.youtube.com/watch?v=7R6172zeNTU` for a full video of this tracking sequence, also accompanied with this paper submission. The tracking ultimately ended because the tool was taken out of the body, and hence, could no longer be seen to track. Because of the track database being updated online, no drift occurs. In addition, because of the estimation of an affine transformation against the database using image metrics that normalize the lighting conditions for feature

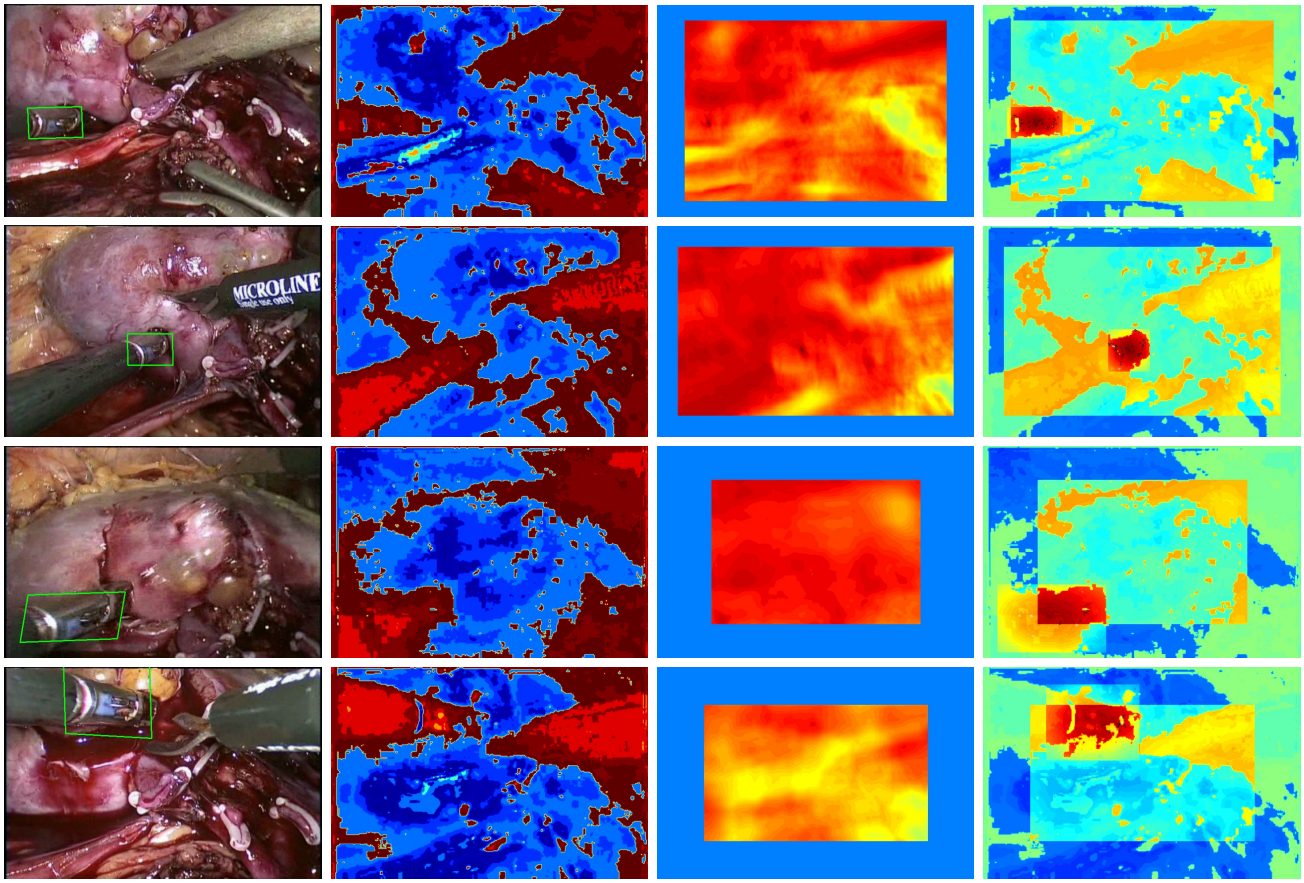

Fig. 4. Four screenshots of tracking a surgical tool in a video sequence from a partial nephrectomy. For each of the likelihoods, red pixels indicate high probability values while blue pixels indicate low probability values. **(Column 1)** The tracked frames, with a green box drawn around the tool as final output from the tracker. **(Column 2)** The color feature likelihood images. **(Column 3)** The correlation surfaces (the blue borders are due to padding outside the window). **(Column 4)** The combined likelihoods (from weighted sums of the individual probability maps) used in the region growing procedure. We show results through severe pose changes, changes in scale, and partial occlusions. In addition, the background is changing as a bleed occurs (bottom row), but the track remains on target. The total time of tracking was 2 minutes and 17 seconds, and ended when the tool was taken out of the body, and hence could no longer been seen by the camera.

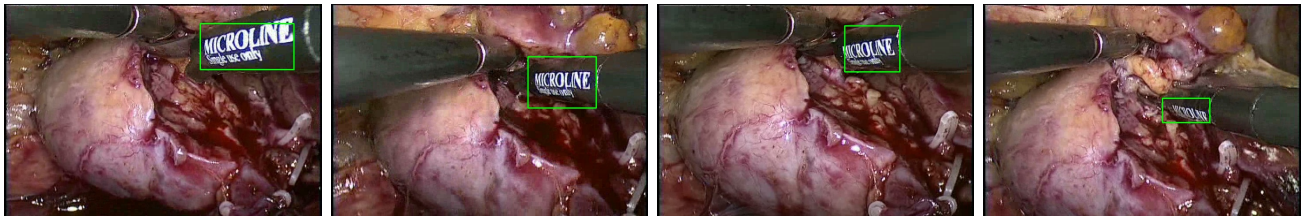

Fig. 5. Tracking the label on the scissors during the removal of part of the kidney in the partial nephrectomy sequence. We are able to capture size and perspective changes as well as partial occlusions.

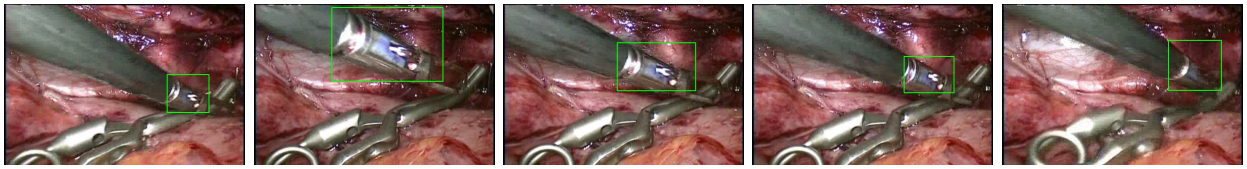

Fig. 6. Tracking the same surgical tool as in figure 4, showing the tracker's ability to persist through significant size changes. The tool starts out (**Column 1**) small and slightly occluded. It then moves rapidly towards the camera and increases in size (**Column 2**), and then goes back to the same location at the original size (**Columns 3-5**), and all of these changes are accounted for.

matching, we are both able to capture geometric changes due to scale and rotational warps as well as changes due to the illumination settings of the endoscope, on a frame-to-frame basis. This significantly increases our tracking abilities over longer periods of time.

A second result, shown in figure 5, uses Feature-Flow to

track the label on the shaft-end of the scissors to cut out part of the kidney during the partial nephrectomy sequence. The initial nomination (left-most column) is perspectively and scale-wise different from columns 2 and 3, yet we are still able to keep track. Column 4 (right-most) shows some rolling of the scissor shaft, resulting in partial occlusion of the label, but the tracker is able to avoid being lost.

As a demonstration of the tracker's ability to adjust to size changes, we show figure 6, tracking the same surgical tool as show in figure 4, but later on in the video sequence. The tool starts off small and slightly occluded (column 1), and then rapidly moves towards the camera, growing significantly in size (column 2). It then goes back to the original location at the original size (columns 3-5), and all of these are accounted for through the full-scale feature matching and geometric warping against the track database.

VI. Conclusion and Future Work

In this paper, we described a tracking algorithm framework using online learning and multiple features working together to track a surgical tool. We show our results on real, in-vivo surgical data where the tracking environment is inherently difficult due to rapid movements, severe occlusions, and viewpoint changes. We stress the importance of being able to discover new information online, using discriminate features, in order to extend the tracker performance over time.

We are currently integrating the tracker with the camera control system of the IREP to perform automated visual servoing tasks [25]. Future work includes reserch into more intelligent database schemes, possibly trying to represent the object spatially (i.e., by 3-D viewpoint) rather then temporally. Also, we are currently investigating an automated method to compute the weights on the individual feature likelihood maps for fusion, based on filter response. We are also looking into methods of selecting representative features of the object when we add it to the database to reduce processing time when matching against the best track state, while retaining the qualitative behavior described in this paper.

VII. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Jaime Landman of New York-Presbyterian Hospital for providing us with human in-vivo data. In addition, Professor Nabil Simaan, Dr. Kai Xu, Dr. Jienan Ding, and Roger Goldman provided valuable insight and assistance with the IREP surgical robot.

References

[1] J. Romanelli and D. Earle, "Single-port laparoscopic surgery: An overview," *Surgical Endoscopy*, vol. 23, no. 7, pp. 1419–1427, 2009.

[2] T. Hu, P. Allen, N. Hogle, and D. Fowler, "Insertable surgical imaging device with pan, tilt, zoom, and lighting," in *IEEE Intl. Conf. on Robotics and Automation*, 2008, pp. 2948–2953.

[3] K. Xu, R. Goldman, J. Ding, P. Allen, D. Fowler, and N. Simaan, "System design of an insertable robotic effector platform for single port access (spa) surgery," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2009.

[4] C. Doignon, F. Nageotte, and M. de Mathelin, "Segmentation and guidance of multiple rigid objects for intra-operative endoscopic vision," in *ECCV 2006, 9th European Conference on Computer Vision*, 2006, pp. 314–327.

[5] S. Voros, J. Long, and P. Cinquin, "Automatic detection of instruments in laparoscopic images: A first step towards high-level command of robotic endoscopic holders," *The International Journal of Robotics Research*, vol. 26, no. 11-12, pp. 1173–1190, Nov 2007.

[6] Z. Pezzementi, S. Voros, and G. Hager, "Articulated object tracking by rendering consistent appearance parts," in *In: IEEE International Conference on Robotics and Automation*, 2009.

[7] M. Groeger, K. Arbter, and G. Hirzinger, "Motion tracking for minimally invasive robotic surgery," *Medical Robotics, I-Tech Education and Publishing*, pp. 117–148, 2008.

[8] G. Q. Wei, K. Arbter, and G. Hirzinger, "Automatic tracking of laparoscopic instruments by color coding," in *Proceedings of the First Joint Conference on Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, 1997, pp. 357–366.

[9] ——, "Real-time visual servoing for laparoscopic surgery," *IEEE Engineering in Medicine and Biology Magazine*, vol. 16, no. 1, pp. 40–45, 1997.

[10] X. Zhang and S. Payandeh, "Application of visual tracking for robot-assisted laparoscopic surgery," *Journal of Robotic Systems*, vol. 19, no. 7, pp. 315–328, 2002.

[11] C. Lee, Y. F. Wang, D. Uecker, and Y. Wang, "Image analysis for automated tracking in robot-assisted endoscopic surgery," in *Proceedings of 12th Intl. Conf. on Pattern Recognition*, 1994.

[12] C. Doignon, F. Nageotte, and M. de Mathelin, "Detection of grey regions in color images: application to the segmentation of a surgical instrument in robotized laparoscopy," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[13] C. Doignon, P. Graebling, and M. de Mathelin, "Real-time segmentation of surgical instruments inside the abdominal cavity using a joint hue saturation color feature," *Real-Time Imaging*, vol. 11, no. 5-6, pp. 429–442, 2005.

[14] A. Krupa, J. Gangloff, C. Doignon, M. de Mathelin, G. Morel, G. Morel, J. Leroy, and L. Soler, "Autonomous 3-d positioning of surgical instruments in robotized laparoscopic surgery using visual servoing," *IEEE Transactions on Robotics and Automation, Special Issue on Medical Robotics*, vol. 19, no. 5, pp. 842–853, 2003.

[15] P. Mountney and G. Z. Yang, "Soft tissue tracking for minimally invasive surgery: Learning local deformation online," in *In: Proceedings of the 11th international conference on Medical Image Computing and Computer-Assisted Intervention*, 2008.

[16] R. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1631–1643, 2005.

[17] Z. Yin and R. Collins, "On-the-fly object modeling while tracking," in *In: IEEE Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[18] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London*, vol. 207, no. 1167, pp. 187–217, 1980.

[19] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *IEEE International Conference on Computer Vision*, vol. 2, October 2005, pp. 1508–1511.

[20] 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*, vol. 24, no. 6, pp. 381–395, 1981.

[21] P. H. S. Torr and D. W. Murray, "The development and comparison of robust methodsfor estimating the fundamental matrix," *International Journal of Computer Vision*, vol. 24, no. 3, pp. 271–300, 1997.

[22] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Intl. Joint Conf. on Artificial Intelligence*, vol. 81, 1981, pp. 674–679.

[23] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[24] J. L. Barron, S. S. Beauchemin, and D. J. Fleet, "On optical flow," in *6th International Conference on Artificial Intelligence and Information-Control Systems of Robots*, 1994, pp. 3–14.

[25] T. Hu, P. Allen, T. Nadkarni, N. Hogle, and D. Fowler, "Insertable stereoscopic 3d surgical imaging device with pan and tilt," in *IEEE Intl. Conf. Biomedical Robotics and Biomechatronics*, Oct. 2008, pp. 311–316.