Noname manuscript No.
(will be inserted by the editor)

Articulated Surgical Tool Detection Using Virtually-Rendered Templates

Austin Reiter · **Peter K. Allen** · **Tao Zhao**

Received: date / Accepted: date

Abstract
Purpose We propose a system capable of detecting articulated surgical instruments without the use of assistive markers or manual initialization.
Methods The algorithm can provide 3D pose using a combination of online and offline learning techniques along with prior geometric knowledge of the tool. It uses live kinematic data from the robotic system to render nearby poses *on-the-fly* as virtual images and creates gradient orientation templates for fast matching into the real image. Prior appearance models of different material classes and projective invariance are used to reject false positives.
Results Results are verified using in-vivo data recorded from the da Vinci® robotic surgical system. The method detects successfully at a high correctness rate and a pyramid search method is proposed which reduces a brute-force method from 23 secs/frame down to 3 secs/frame.
Conclusion We have shown a top-down approach to detect surgical tools within in-vivo video sequences and is capable of determining the pose and articulation by learning *on-the-fly* from virtual renderings driven by real kinematic data.

A. Reiter, P. K. Allen
Dept. of Computer Science
Columbia University
500 W. 120th Street, M.C. 0401
New York, NY 10027
Tel.: +1 212-939-7093
Fax: +1 212-666-0140
E-mail: areiter, allen@cs.columbia.edu

T. Zhao
Intuitive Surgical, Inc.
1266 Kifer Road, Bldg. 101
Sunnyvale, CA 94086
Tel.: +1 408-523-2100
Fax: +1 408-523-1390
E-mail: tao.zhao@intusurg.com

Keywords Medical Robotics · Tool Tracking · Computer Vision · da Vinci

1 INTRODUCTION

Despite the introduction of minimally-invasive techniques to general surgery more than 20 years ago, many of the more advanced surgical procedures are still too difficult to perform laparoscopically and require a steep learning curve. Limited dexterity and reduced perception have limited surgeon's abilities to achieve comparable results to open surgery. The introduction of tele-operated master-slave robots such as the da Vinci® surgical system from Intuitive Surgical, Inc. [1] has alleviated many of these issues by offering dexterous manipulation through more than 40 different types of articulated tools, binocular vision, and a separated surgeon console with good ergonomics.

Robotic technology can enhance the capabilities of a surgeon, for example, in the case of motion scaling which enables more finer-scaled manipulations than may be possible with the human hand, currently done by the da Vinci®. As the research community continues to develop intelligent algorithms, more skills are put into the hands of the surgeon, and better techniques get developed as a result which help improve the outcome of surgeries worldwide.

Surgical tool tracking is one such example which provides enhanced situational awareness. Knowing the locations or the full poses of tools in the endoscopic video can enable a wide spectrum of applications. Virtual measurement capabilities can be built upon it which provides accurate measurement of sizes of various anatomical structures. Virtual overlays indicative of the status of the tool (e.g., the firing status of an electro-cautery tool) can be placed on the instrument shaft at the tip of the instrument which is close to the surgeon's visual center of attention, enhancing the safety

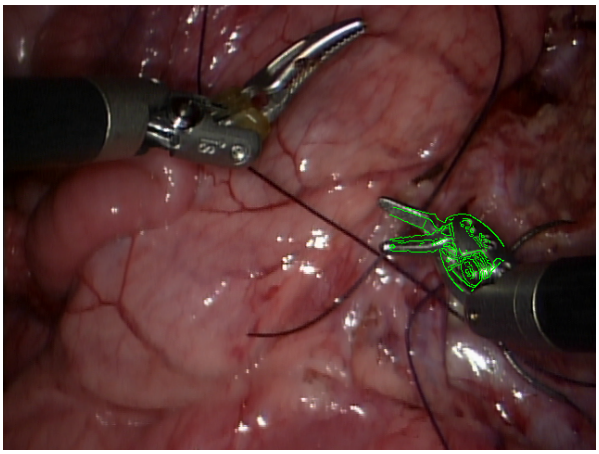

Fig. 1 Our tool detection algorithm both automatically finds where in the image the tool exists as well as the pose and articulation according to the kinematic joint configuration. Virtual renderings which are driven by kinematic data are constructed *on-the-fly* and used to find the best matching joint configuration based on the observation from the video frame. The green overlay shows the corrected kinematic configuration according to our matching methodology.

of using such tools. The knowledge of the tool locations is also useful in managing the tools that are off the screen, increasing patient's safety, or can be used for visual servoing to expand the effective visual field of view, in the case of motorized camera systems.

The joints of a robotic surgical system are usually equipped with sensors (e.g., encoders) for control. The pose of the end effectors can be computed using forward kinematics. Therefore, the poses of the tools in the camera frame are readily available, in theory. The kinematics chain between the camera and the tool tip involves 18 joints and more than 2 meters in cumulative length, which is challenging to the accuracy of absolute position sensing. However, a master-slave system such as the da Vinci® does not require high absolute accuracy because surgeons are in the control loop. As a result, we have observed up to 1-inch of absolute error, which is too large for most of the applications that are mentioned above. Therefore, detecting and tracking the tools from images is a practical way to achieve the accuracy requirements of the applications.

Surgical tool tracking can range from 2D patch tracking (e.g., identifying and following a 2D bounding box around the tool in a video sequence) [2] to 3D pose tracking (e.g., recovering the full 6-DOFs of rigid motion) [3] [4]. Tool tracking approaches typically divide into two categories: using *markers* or *fiducials* to assist in localization [5] [6] [7], and **marker-less** approaches which use natural information [8] [9]. Often the task is to find the tool by its long, tubular shaft [3], however in our experience many da Vinci® surgeons prefer to work with the camera very zoomed-in so that only a small part of the shaft is visible, discounting its value in detection. Information may also be added to the scene from an additional source [10] to determine the pose of the instrument using prior knowledge of the source. Template matching has also previously been applied to surgical tool tracking [11] [12].

As tools become more dexterous and articulation is included, pose tracking becomes more complicated. Although there are typically kinematic models available to describe this motion, they tend to be too inaccurate to perform precise localization. The work in [13] is inspiring where articulated 3D models are matched to the mask from an image segmentation procedure using a binary silhouette. Our work solves a similar problem, but we make use of a different kinematic template matching method which uses gradient information to match to the finer texture and shape of the tool tip.

2 OVERVIEW

In this work, we strive to do away with using customized markers to aid in tool tracking and we wish the method to work robustly in real in-vivo environments. As such, we make use only of the natural visual information which is available on the tools and in the scene to determine a full 3D pose with articulation. We combine raw kinematic data with both on-line and off-line learning techniques. We use a virtual renderer driven by a kinematic model consistent with a tool on the robot to render the tool according to raw kinematic estimates. By perturbing the tool in a small range of joint values around the kinematic estimate, we can re-render the tool at each potential joint configuration and extract templates to match against the real video imagery. The template which matches best to the image corresponds to the corrected joint kinematics.

For purposes of this work there are 2 goals: (1) where in the image is the tool? and (2) what is the pose (including articulation) of the tool? In the computer vision community, algorithms are typically either *top-down* or *bottom-up*. A top-down approach to solving this problem would be to match the entire object as a whole where all information is used at once, although this tends to be quite computationally-expensive with a potentially large search space (6-DOFs for rigid motion and up to 3-DOFs or more for articulation). On the other side is the bottom-up approach, which would be more feature-based. Here, we would start from smaller features and "build-up" applying geometric constraints later on in the algorithm [14]. Although appearance variation is smaller at the feature level, and thus easier to model and faster to compute, they tend to be less discriminative than an entire pattern. In this work, we apply template matching as a top-down approach to the pose estimation problem.

The basic idea to applying template matching to the pose estimation problem is as follows:

– Store templates corresponding to different poses
– Match all templates against the image and find the *best match location* for each template in the image
– Sort the template responses and take that which responded best to the image
– Recover the corresponding pose for that template

We construct the templates using a robot renderer which uses graphical techniques to create virtual images of the tool from a CAD model according to specific kinematic joint configurations. This is desirable because collecting training data becomes easier than if it had to come from video ground truth and so we can collect more of it with less effort. The advantages of this type of data generation has been shown successfully in [15]. Typical template matching uses luminance images, however this is generally not photo-realistic enough to match against real imagery [16][17]. Recent work [18] (called LINE) has introduced a novel image representation which stores discretized gradient orientations at each pixel as a compact bit-string. Each template then consists of high-contrast spatially-located gradient responses which represent a particular configuration of the object. A novel similarity measure which is able to maximize the cosine of a difference of gradient orientations in a small neighborhood using a lookup-table optimized to standard memory caches allows for real-time matching of many templates to a single image. Then, the template matching amounts to comparing gradient orientations at each pixel rather than luminance values. In this way, we can create templates from virtual renderings which contain information that can be successfully matched into a real image of the same object. We refer the interested reader to [18] for more details on the LINE algorithm.

An advantage of using virtual renderings to create training data is that it is easily extensible to many different tool types. Rather than requiring an arduously-manual procedure for creating training data for a specific tool (off-line), training amounts to a CAD model and a renderer (on-line). The major contribution of this work is demonstrating the ability to successfully match virtual renderings robustly to real imagery, which in the past has only been done effectively with depth imagery. In addition, by generating image templates through a kinematic model, we are able to tie joint configurations to appearance changes and efficiently recover the pose through a direct image match. We note that although this method is presented for use on the da Vinci® surgical system, the approach is general to any robotic arm where templates can be virtually-rendered by kinematic data and matched into an image frame.

2.1 Virtual Renderings

We begin with the image frame from the endoscope of the da Vinci® robot. In this work, although stereo imagery is available only one of the cameras is necessary as this is a model-based approach. Each frame has synchronized kinematic data for all parts of the robot through an application programming interface (API) [1]. Figure 2 shows a sample video frame from the left camera of the stereo endoscope as well as the virtual rendering corresponding to the kinematic estimate for that frame. Notice that although the estimate produces a close approximation to the appearance of the tool, it is inaccurate. Note that we only use the tool tip and hide the clevis and shaft for simplicity. In our experience, this was sufficient information to perform successful matching.

2.2 Matching

Using the virtual CAD model and the raw kinematics from a given frame, we re-render the model at nearby poses. By perturbing only a few of the active joints, we can generate renderings *on-the-fly* to match against the real image and take the kinematics estimate corresponding to the best matched template. For our experiments, we use a Large Needle Driver (LND) of the da Vinci® which has 7 total DOFs. We chose to perturb 5 of the 7 active joints, corresponding to some of the pitch and yaw parameters of the wrist and instrument as well as the roll of the instrument. However, this method allows for perturbations of whichever joints provide significant errors. We cycle each of the joints in a small range around the raw kinematics estimate and re-render the CAD model in that pose. For each, we construct a gradient tem-

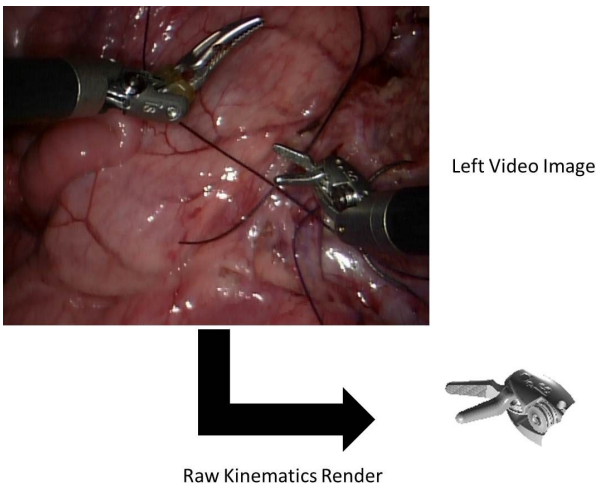

Fig. 2 A sample image from the endoscope and the corresponding virtual render from the raw kinematics. Although the configuration is close, it's visibly-imprecise from what is observed in the video frame.

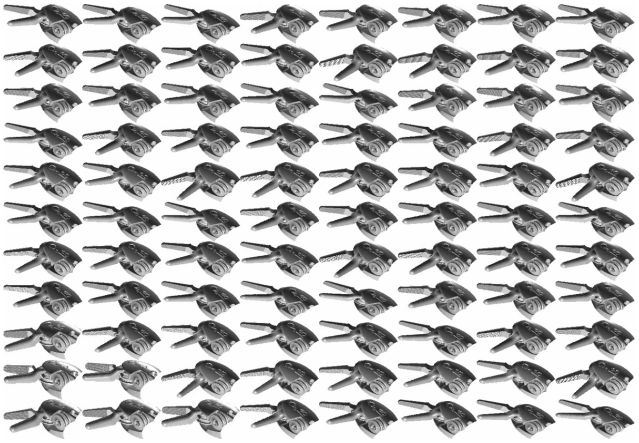

Fig. 3 A sampling of 88 out of the total 1125 virtual renderings from the brute-force joint search matching procedure.

plate using the LINE detector and store meta-data for the kinematics with each template. Once collected, we compute a similarity measure (according to [18]) of each gradient template against the real image. For each gradient orientation in the template, a search is performed in a neighborhood of the associated gradient location for the most similar orientation in the real input image. The maximum similarity of each template to the image is stored and sorted across all templates to find the best matching gradient template(s) corresponding to the *corrected kinematics* estimate.

2.2.1 Brute-Force Search

To perturb the joint values, we manually define a bounding box around the kinematic estimate as follows:

- Outer-yaw: $\pm 1.5°$ every $1.5°$
- Outer-pitch: $\pm 1.5°$ every $1.5°$
- Instrument-roll: $\pm 10°$ every $5°$
- Wrist-yaw: $\pm 15°$ every $7.5°$
- Wrist-pitch: $\pm 15°$ every $7.5°$

This range results in 1125 total renderings, each to be compared against the video frame. A sampling of 88 of these virtual images, displaying only the tool tips, is shown in Figure 3. Figure 1 shows the result of this search using LINE templates of each render candidate to both find where in the image the tool tip for the LND is located as well as the joint configuration corresponding to the best matching template response. The overlay is produced by only showing the edges (in green) of the best-matching template (from the virtual rendering), so it's easier to observe the accuracy of the alignment qualitatively.

In practice, the best-matching template according to the LINE search is not necessarily always the correct answer. To account for this, we collect the top k (~ 5) best matches. It can often occur that the actual correct response is the 2^{nd} or even 3^{rd} best LINE match. This often occurs when the top matches have the same score and are arbitrarily ordered. In these cases, what we noticed is that the better-scored matches occur elsewhere in the image, either on the other tool or on the background tissue. However, this is something that we can account for, as described in the next section.

2.3 False Positive Rejections

In order to increase the detection rate, we collect more of the top matches from the LINE responses. Because some of the higher responses may sometimes fall in other areas of the image (Figure 4, left), we developed two false positive rejection schemes.

2.3.1 Likelihood Rejection

Using the method described in [13] (and code generously provided by the authors) we train off-line from manually-labeled sample video images to construct class-conditional probabilities using several color and texture features at the pixel level. We train using 3 classes to label each pixel as either on the tool shaft (class 1), metal (class 2), or tissue (class 3). The data which was used to train was from a completely separate data set than what was used for testing, to further confirm the usefulness of the method. Pixels that are labeled as metal correspond to the tool tip or clevis and allow us to filter false positives by rejecting those LINE detections which fall in probabilistically-low regions in the metal likelihood labeling. This threshold is determined empirically, however future work plans on investigating more automated methods of determining this threshold. One such possibility is to analyze a histogram of probability values and comparing to a prior based on the expected number of pixels on the tool tip and clevis.

An example of the output of this training procedure is shown in Figure 5. The upper-left image shows the original

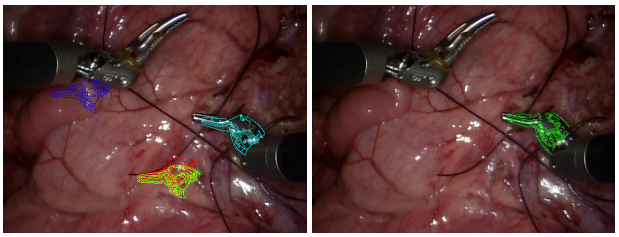

Fig. 4 Using our likelihood rejection method, we can reject false positives [**Left**] and recover the best matched pose [**Right**] with a higher detection rate by collecting more of the top matches and eliminating those that don't probabilistically fall on the tool tip. Green labels the best match according to LINE, which is clearly off the tool in the left figure. However, after applying the false positive rejection method on the right, the top match is more visibly consistent.

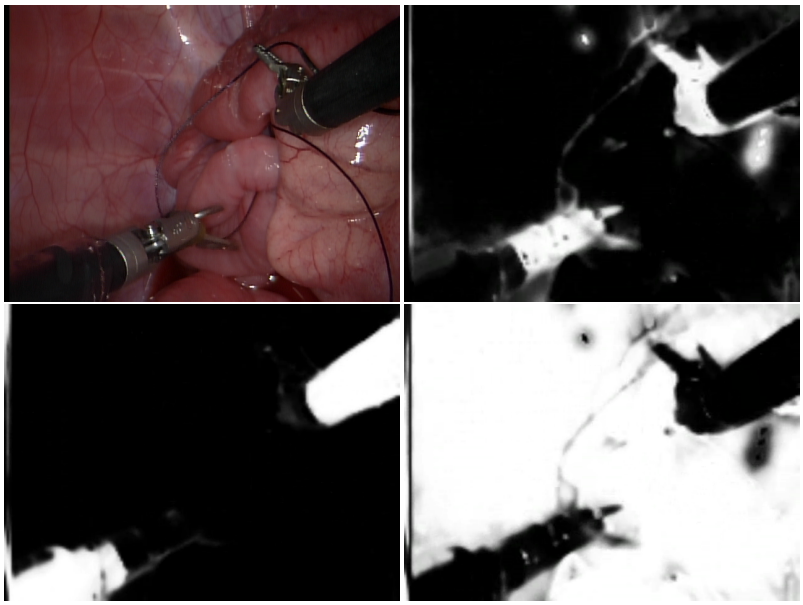

Fig. 5 Example likelihood images from class-conditional pixel labeling after the training procedure described in section 2.3.1. [**Upper-Left**] The original image from an in-vivo sequence of 2 LND tools performing a suturing procedure. [**Upper-Right**] Metal likelihood (*e.g.*, tool tip, clevis) [**Lower-Left**] Tool shaft likelihood [**Lower-Right**] Background class likelihood

image frame from an in-vivo sequence containing two LND tools performing a suture. The upper-right is the *metal* class likelihood labeling, where brighter pixels indicate higher probabilities. The lower-left image similarly shows the *shaft* class likelihood labeling, and finally the lower-right shows the *background* class. As you can see, the metal labeling sufficiently marks pixels in the area where the tool detections should occur, and helps reduce false positives in other areas. The shaft labeling can also be put to use, as described in the next section.

2.3.2 Vanishing Point Rejection

Sometimes a false positive can fall on the other tool in the image on the metal tip, where the previous rejection method would fail. To further reduce false positives, we noticed the following: the parallel lines that make up the edges of the tool's shaft boundary share a vanishing point (VP) in common. Furthermore, the shaft of the tool will always come into the image frame from the edge of the image. We can compute the VP according to the shaft orientation from the kinematics and the camera model [19]. For each of the LINE hypotheses (each of which has a candidate kinematic joint configuration), we can identify the corresponding end of the proximal clevis (the metal part which connects the black shaft with the metal tool tip, at the wrist) as an arc consistent with that hypothesis. Then we connect the VP and the two ends of the proximal clevis arc and extend the lines to the image boundary. Figure 6 describes this idea visually. We can obtain the regions A (red) and B (green) from the proximal clevis arc and compute the overlap of those regions with the shaft likelihood labeling, described in section 2.3.1. Then, region B is easily rejected as a false positive while region A persists as a possible candidate match.

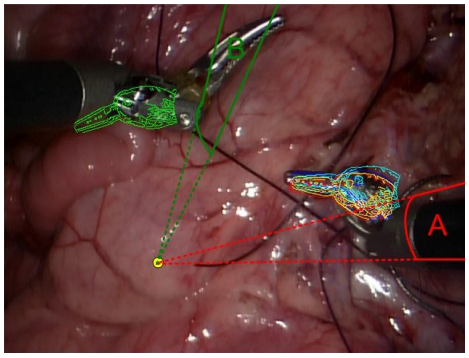

Fig. 6 We compute the vanishing point of the tool shaft we are detecting using kinematics data and a camera model. Then we connect this with the candidate proximal clevis of each of the hypotheses according to the top LINE matches and use the shaft likelihood labeling to eliminate projectively-inconsistent kinematic hypotheses. The region labeled A in red marks the proximal clevis associated with the red LINE detection on the proper tool tip. The region labeled B in green is a false positive on the wrong tool. By insersecting these regions with the shaft likelihood labeling from section 2.3.1, region A intersects many pixels with the shaft likelihood while region B has a null intersection.

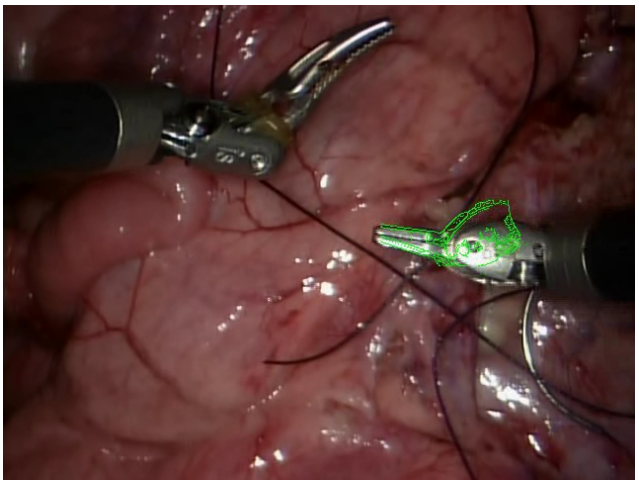

Fig. 7 A failure case is defined as a detection that occurs on the correct tool tip with a visibly-wrong joint configuration.

3 EXPERIMENTS

Experiments were run on porcine in-vivo data from the da Vinci® surgical system. In the following sections, we discuss accuracy and timing results from our experiments.

3.1 Accuracy

The main failure case that we encounter is shown in Figure 7, where the detection occurs on the correct tool tip, but with a visibly wrong configuration. This is determined by eye as a simple counting of overlaying pixels tends to call these correct when in fact they are wrong. All other false positives (e.g., detections that occur on tissue or the other tool) are successfully rejected by our false positive rejection method. In an experiment with real in-vivo data, our brute-force method yielded an 87% correctness rate (on the correct tool tip, in a visibly-consistent joint configuration according to the template overlay). However, we obtained 11% with no detection, meaning all of the top 5 matches were rejected by the false positive method on those frames. This is a preferable result as we then have more confidence in our actual detections and would prefer a *no-response* over a *wrong response*. Finally, 2% were labeled incorrectly, with wrong configurations (determined by eye).

From our analysis, the remaining incorrect detections tend to occur when the tool is in an ambiguous configuration, such as extended straight out or if little texture is shown. The cases that work very well are when the clevis bends at an angle from the shaft, creating a unique shape that's more easily detected by the LINE templates. We plan on investigating this further to improve the detection rate.

3.2 Timing

The brute force method described in Section 2.2.1 renders 1125 templates per-frame, which is not a real-time solution. There are several potential speed-ups which can retain the accuracy described in Section 3.1 while speeding-up the processing per-frame. The tests of each template against the image frame are independent from each other, and as such are certainly parallelizable, either on the GPU or on a multi-core/processor CPU, neither of which we implemented for this paper. Other than that, the two significant bottlenecks are the *number of renderings* as well as the *time per-rendering*. We use OpenGL to render the model in particular configurations, and downloading the image from the GPU into software to process each template is computationally-expensive. Rather than downloading the entire image, we use the kinematic data to determine a region-of-interest in the virtual image where the tool-tip will be drawn and only download that part of the graphics buffer. However, the number of renderings needs to be reduced as well. The brute force search takes $\sim$ 23 seconds/frame to match all templates and produce the best pose. In the next section we describe an approach which yields an **87% speed-up**.

3.2.1 Pyramidal Search

We make the assumption that the manifold of appearances for the tool tip is relatively smooth, and so a very coarse search should yield a match which is close to the actual result. We can then use this to initiate a finer-scaled search in a much smaller range and reduce the overall number of templates to render. We choose a 2-level pyramid search scheme where the actual joint search ranges were determined by hand in both a *coarse* and *fine* manner. The coarse search uses the following search range, relative to the raw kinematics estimate as in the brute-force method:

- Outer-yaw: $\pm 1^\circ$ every 2°
- Outer-pitch: $\pm 1^\circ$ every 2°
- Instrument-roll: $\pm 10^\circ$ every 10°
- Wrist-yaw: $\pm 15^\circ$ every 15°
- Wrist-pitch: $\pm 15^\circ$ every 15°

This yields a total of 108 renders in the first-level. We take the top k-matches using LINE and use the same false-positive rejection method as described in Section 2.3. The template with the best response to survive the false-positive rejection initiates a finer-scaled search with the following search range, relative to the joint configuration which produced the coarse match:

- Outer-yaw: $\pm 0.25^\circ$ every 0.5°
- Outer-pitch: $\pm 0.25^\circ$ every 0.5°
- Instrument-roll: $\pm 5^\circ$ every 5°
- Wrist-yaw: $\pm 5^\circ$ every 5°

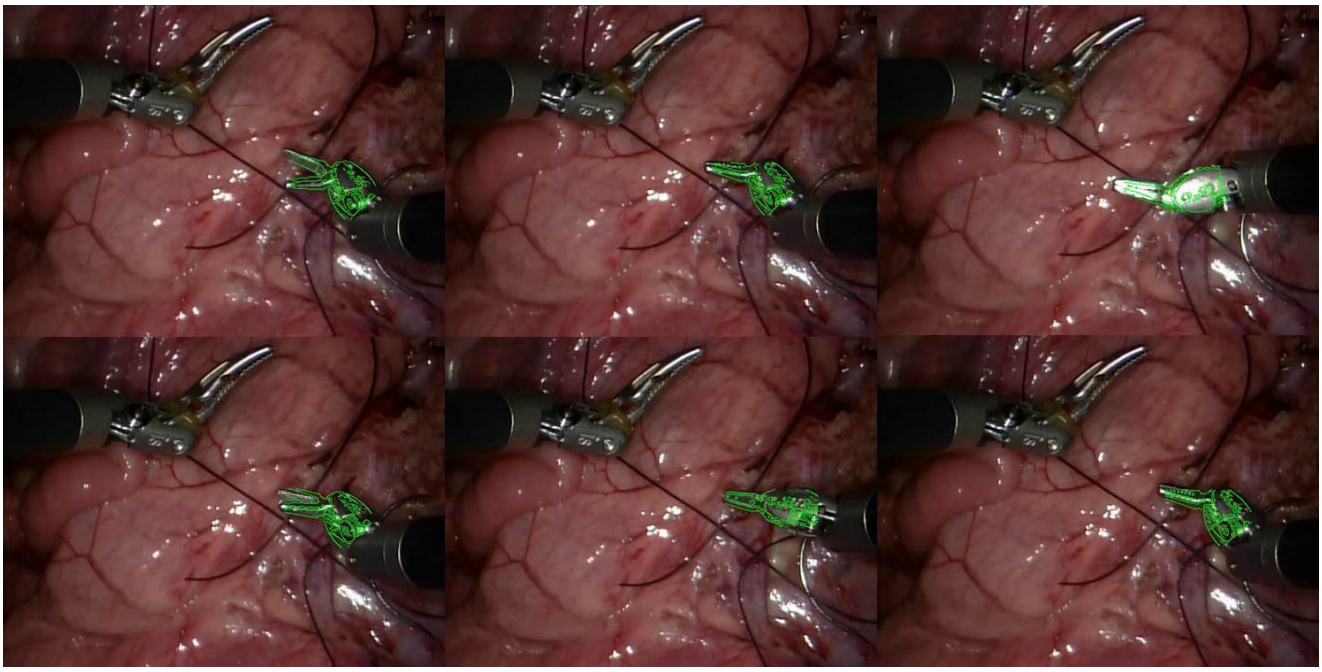

Fig. 8 Sample output frames from running the tool detector on an in-vivo video sequence. Several different poses are shown as well as an example with high specularity (top-right). The green overlay is produced by drawing only the edges of the best-matching virtually-rendered template, to assist in seeing the quality of the result. These results are produced from the pyramid-search method, described in section 3.2.1.

– Wrist-pitch: $\pm 5^\circ$ every 5°

This produces an additional 108 renders for a total of 216 total renderings in the pyramid search, a reduction of 81% in the overall number of renderings per-frame from the brute-force method. Using the same LINE matching and false-positive rejection, we take the joint values corresponding to the best match in the fine-scaled search as the final result. This reduces the total per-frame processing time from 23 secs/frame down to 3 secs/frame, an 87% speed-up in processing time. Figure 8 shows sample frames from running the detector using the pyramid-search method. The green overlays show the edges of the best virtually-rendered templates (again only displaying the edges of the templates which help visualize the quality of the match). We show several different poses and articulations of the tool, as well as an example with significant specularities (upper-right image). The detection rate of the pyramid-search method was nearly identical to the brute-force method, at 86% correctness.

4 DISCUSSIONS

The power of this approach lies in the ability to learn from virtual renderings which are kinematically-generated and accurately apply this to real imagery. This scheme has been shown to be successful in the object recognition [20] and human pose recognition [15] domains where **depth images** are used to learn appearances. In the case of [15], synthetic depth images render articulated human poses to create a richer database which is not feasible to do with the amount of real data needed to accomplish the same task. In our work, we provide a means to accomplish a similar goal for matching against **real imagery** by use of a template which is created from a virtual image and is matchable against a real image.

The consequences of this are two-fold: first, we are able to learn *on-the-fly* without knowledge of the poses that can occur during the surgery, and thus can reduce the memory footprint of a database-like approach which must account for all possible poses and articulations to succeed robustly. Second, the method becomes easily extensible to different tool types. As new tools are developed for a robot such as the da Vinci®, we can easily apply our algorithm to it in real imagery with nothing but a CAD model and raw kinematics estimates. Another advantage is that the false rejection method is not specific to any tool-type, but rather is material-specific and can be applied to different tool types directly.

Being a top-down method for a high-dimensional problem, computation is usually a bottle neck. There are multiple ways to further reduce the computation time to bring it down to real-time. Since rendering is already done on the GPU, the run-time can be pushed down further if the generation and matching of the LINE templates are also done on the GPU, both taking advantage of parallelization and avoiding the large memory transfer from the GPU. Gradient-based search routines, such as Levenberg-Marquardt, could poten-

tially reduce the matching time although it may not find the optimal solution. The accuracy of any optimization routine must be considered as well. We also note that the likelihood maps described in Section 2.3.1 may have a further use to reduce the number of renderings. This pixel labeling can cue a data-driven joint search, where the search range can be potentially determined using inverse kinematics of only those configurations that are consistent with the pixel labeling of the metal-class (e.g., tool tip and clevis). This is an area for future exploration.

Although we investigate the detection problem in this paper, the final system will run in a tracking mode. We notice that between consecutive frames in a video sequence there is only very little movement that tends to occur. This means that the set of virtual renderings will only need to be updated slightly, and most of the templates can be reused from frame-to-frame. This should also help to increase the detection accuracy rate by confining searches to previous successfully detected locations. We then plan to fuse this into a live experiment to compute a kinematics correction on-line and evaluate the effects of the enhancements of some of the applications previously mentioned to a surgeon's experience.

5 CONCLUSIONS

In this work we have presented a detection method which uses a top-down approach to detect surgical tools in video sequences and determines the pose and articulation by learning *on-the-fly* from virtual renderings driven by real kinematic data. We have shown that the method detects successfully at a high correctness rate and there is much room for improving the runtime. We showed a pyramid search method which reduced a brute-force method from 23 secs/frame down to 3 secs/frame. This may be reduced further by using more sophisticated data-driven optimization routines and using GPU accelerations to eliminate downloading each render for each potential joint configuration.

References

1. "Intuitive Surgical, Inc." http://www.intuitivesurgical.com/. 1, 3
2. A. Reiter and P. K. Allen, "An online approach to in-vivo tracking using synergistic features," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, October 2010. 2
3. 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. 2
4. C. Doignon, F. Nageotte, and M. de Mathelin, "Segmentation and guidance of multiple rigid objects for intra-operative endoscopic vision," in *European Conf. on Computer Vision*, 2006, pp. 314–327. 2
5. M. Groeger, K. Arbter, and G. Hirzinger, "Motion tracking for minimally invasive robotic surgery," in *Medical Robotics, I-Tech Education and Publishing*, 2008, pp. 117–148. 2
6. G. Q. Wei, K. Arbter, and G. Hirzinger, "Automatic tracking of laparoscopic instruments by color coding," in *Proceedings of the First Joint Conf. on Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery*, 1997, pp. 357–366. 2
7. 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. 2
8. C. Lee, Y. F. Wang, D. Uecker, and Y. Wang, "Image analysis for automated tracking in robot-assisted endoscopic surgery," in *Proceedings of the 12th Intl. Conf. on Pattern Recognition*, 1994. 2
9. 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 *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2004. 2
10. 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. 2
11. G. D. Hager and P. N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, 1998. 2
12. D. Burschka, J. J. Corso, M. Dewan, W. Lau, M. Li, H. Lin, P. Marayong, N. Ramey, G. D. Hager, B. Hoffman, D. Larkin, and C. Hasser, "Navigating inner space: 3-d assistance for minimally invasive surgery," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2004. 2
13. Z. Pezzementi, S. Voros, and G. Hager, "Articulated object tracking by rendering consistent appearance parts," in *IEEE Intl. Conf. on Robotics and Automation*, 2009. 2, 4
14. I. Gordon and D. Lowe, "What and where: 3d object recognition with accurate pose," in *Toward Category-Level Object Recognition*, 2006, pp. 67–82. 2
15. J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011. 3, 7
16. G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter sensitive hashing," in *IEEE Intl. Conf. on Computer Vision*, 2003. 3
17. H. Ning, W. Xu, Y. Gong, and T. S. Huang, "Discriminative learning of visual words for 3d human pose estimation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2008. 3
18. S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *IEEE Intl. Conf. on Computer Vision*, 2011. 3, 4
19. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003. 5
20. R. Ohbuchi, K. Osada, T. Furuya, and T. Banno, "Salient local visual features for shape-based 3d model retrieval," in *IEEE Intl. Conf. on Shape Modeling and Applications*, 2008. 7