Low Resolution Tool Tracking for Microsurgical Training in a Simulated Environment

Antonio Carlos Furtado, Irene Cheng, Eric Fung, Bin Zheng and Anup Basu

***Abstract*— In this work, we propose a method that detects and tracks the tip of tools used in microsurgical training. This method can be used to provide valuable metrics regarding the surgeon's hand movement. It can benefit the training of surgeons, given the steep learning curve in microsurgery. Unlike past research, our tool tracking algorithm does not rely on color based measurements. Thus, it can be used in a broader domain. Also, our approach is robust to surrounding environments with non-static background, where background subtraction techniques are not suitable. Experimental results show that the proposed tool localization method has high accuracy and is statistically reliable.**

I. INTRODUCTION

Microsurgery has evolved from suturing blood vessels to performing more complex procedures, and extension to robot-assisted microsurgery. Despite the technological advancements, a steep learning curve, costly set-up, restricted area of vision, loss of depth perception and loss of visual reference points[1], [2] remain challenging in surgical training. The current model of training mainly relies on subjective measures, which are prone to multiple biases and translate into extra hours of work.

Objective assessment has gained considerable interest in recent years. Eye and hand movement metrics are the most widely used performance indicators of surgical expertise[3], [4], [5]. While technology advancements in eye tracking sensors have made it easier to obtain eye-related metrics, hand movement data is still commonly estimated through tool tracking techniques. Multiple tool tracking methods have been proposed in the past, most of them focus on laparoscopic surgical environments[6], [7], [8], [9]. However, current techniques have certain limitations, since many of them have special requirements for the experimental setup, such as the placement of color markers on tools[6], use of multiple cameras[9], availability of graphics cards[8] or having a mostly static background[7].

In this paper we propose a tool detection and tracking strategy which can locate the tool position for a simulated suture task. Our method does not require any special equipment setup during the experiments. This means that the setup does not interfere with the actual experiment. Furthermore, we demonstrate that our method works with low resolution videos, and is robust to visual artifacts encountered while record the video of the procedure. As the goal of this method is to aid analysis of tool movement, no online usage for this method is suggested. For this reason, this paper focuses of the accuracy of the method, rather than computational costs.

A.C.F., I.C. and A.B. are with the Department of Computing Science, University of Alberta, Edmonton, T6G 2R3, Canada (e-mail: basu@ualberta.ca)

E.F. and B.Z. are with the Department of Surgery, University of Albert, Edmonton, T6G 2R3, Canada (e-mail: bzheng1@ualberta.ca)

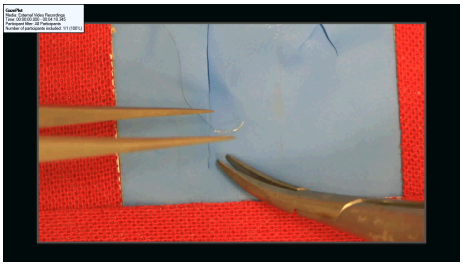

Fig. 1. Simulated suture environment showing the needle with a line going through the simulated skin, a needle driver on the right, and an auxiliary tool on the left.

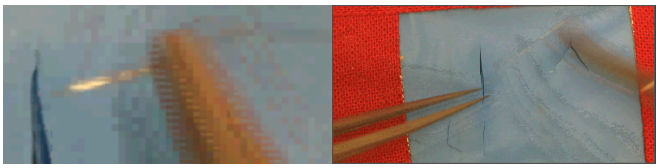

Fig. 2. Illustration of visual artifacts encountered in a simulation sequence. On the left side, we demonstrate the occurrence of motion blur. On the right side, we present a frame that contains ghosting. These two are the major artifacts found.

II. METHOD

The tool tracking method proposed by this work was applied on a simulated suture experiment. The complete procedure was recorded, by placing a camera above the training pad, in a parallel position. The video was later used to extract tool tracking metrics. The surgical environment is illustrated in Figure 1.

Accurate tool tracking is paramount for performance evaluation in order to build the evaluation metrics. In this study, our goal is to track the needle driver used to grasp and guide the needle. This goal turns out to be challenging, given the visual artifacts encountered in the recorded video (illustrated in Figure 2). Our tracking technique needs to overcome these artifacts. It should also handle situations where the object may leave and enter the scene multiple times. For this reason, our tracking method requires a detection mechanism that is able to identify the object at any given frame, or when it first appears in the video. The tracking mechanism also serves to estimate the displacement of the object between frames.

The object detection approach adopted for the needle driver makes some assumptions about how the tool is positioned, and how it is oriented. These assumptions are

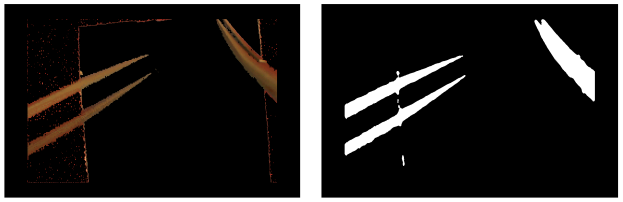

Fig. 3. Illustration of the process used to extract silhouettes that represent surgical tools. On the left image, threshold is applied to hue and saturation. On the right image, thresholded image is followed by a median filter.

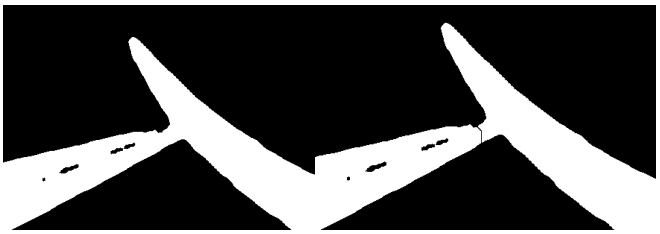

Fig. 4. Sequence illustrating the effect of segmenting blobs through watershed segmentation. On the left image, blobs are initially connected. On the right image, watershed segmentation is applied to blob.

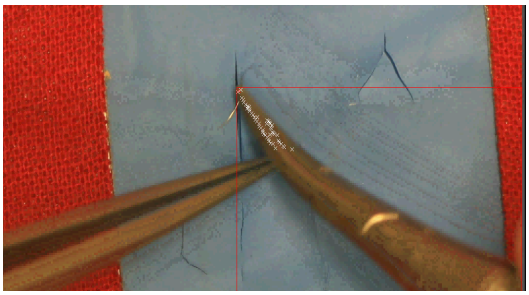

Fig. 5. Harris corner detection is applied to a frame. Corners are filtered by using tool blob area, and only corners inside the needle driver are selected. Tool tip position is represented by the red point.

based on how the experiments were conducted. The first assumption is that the needle driver is always operated by the right hand. Thus, we can also assume that the tool is north-west oriented. Additionally, given that the same set of tools are used by all the subjects for all the procedures, we can also assume that they possess the same color and texture properties, with some minor lighting variations. Even though microsurgical tools can present minor differences in shape and color, the majority tend to be sharp and monochromatic. Our method can be easily adapted to any other test environments where these conditions are present. However, fine-tuning is required for other experiments.

The detection process is started by applying a color threshold to the image. Here, instead of relying on the Red-Green-Blue (RGB) color space, we convert the frame to the Hue-Saturation-Value (HSV) color space. The HSV color space allows us to isolate the chrominance components (i.e., the hue and saturation) from the brightness component (i.e., the value component), which is not possible in the RGB color space. While other color space standards also allow chrominance isolation, HSV was chosen because of the low computational overhead for conversion. By applying a threshold only to the chrominance components, we are able to make our detection step more robust against lighting variations. The thresholding mask result obtained from an image is illustrated in Figure 3(right). The needle driver can be seen on the right-side of the image.

Observe that both surgical tools have the same color. This is a problem when they intersect each other, as shown in Figure 4. In order to address this issue, we use the Watershed algorithm[10] for segmenting the two blobs. By applying the Watershed algorithm we can isolate each blob, and keep track of the desired tool, which we assume to be held by the right hand. Tools with different colors can still be handled by our method. In this case, HSV filtering thresholds need to be updated accordingly.

After identifying the blob that corresponds to the needle driver, we use its area to extract interest points in the original frame. In the current implementation, we used corners as interest points. Corners are the intersection of two edges. They represent the point at which the direction of these two edges change. Therefore, the gradient of the image in both directions have a high variation, making it easier to detect and track. We detect these corners by using the Harris corner detector[11]. The detector starts by computing the first order derivatives (I_x, I_y) of the image, to highlight the directional intensity variations. A second moment matrix, which encodes this variation, is calculated for each pixel in a small neighbourhood, as follows:

$$Z = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}$$

A patch defined by Z is accepted as a candidate feature if its minimum eigenvalue $\lambda_{\min}$ is above a predefined threshold t_f. Given that the eigenvalues of this matrix are sufficiently large, the patch can be characterized as a corner texture. Therefore, the higher the threshold value, the greater is the gradient intensity difference, meaning that it is a stronger corner candidate. In our case, defining an optimal t_f is not very important because the detected blob already provides a precise estimate of the tool region.

The corner detection applied to the needle driver is illustrated in Figure 5. Note that we do not detect points on the whole needle driver, but only on the area around the tip. We discard the candidate corners at the bottom, as they are not relevant, when estimating the tooltip position, represented by the red point in Figure 5.

Once the corner points are detected, it is still necessary to track them. We use the KLT tracker[12], [13] to perform this task. The objective of the tracker is to compute the translation (du, dv) of a region, centered on our corner point, which is calculated iteratively. This computation is based on an optical flow method. Optical flow methods are used to generate dense flow fields, which are useful in detecting movements. They compute the flow fields by calculating the flow vector of each pixel under the brightness constancy constraint: $I(x, y, t) - I(x + dx, y + dy, t + dt) = 0$. This computation

is done on the neighbourhood of a pixel. Based on the brightness constancy constraint, the KLT tracker computes the displacement of the path using the following formula:

$$\begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{pmatrix} du \\ dv \end{pmatrix} = \begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

where I_t represents the temporal gradient. Once the updated location of the interest point is obtained, the KLT tracker evaluates the quality of the tracked path by using the affine transformation between the consecutive frames. In case the sum of squared differences of the current patch and the projected patch is small enough, they continue tracking the feature. Otherwise, the feature point is discarded. For our experiments, if more than 75% of the corner points are discarded, we rerun the detection method. The detection should provide new corner points to be tracked. If it fails to acquire new points, we assume the tool is not present in the current frame, and the detection method is rerun on the subsequent frames, until new corner points are found.

We acknowledge the fact that the number and position of corner points can vary greatly, depending on the tool orientation. However, based on our ultimate goal, the tool tip is always assumed to be either the top left or the top right corner, and its position is derived from other corner points, as seen on Figure 5. This diminishes the effect of changes in point detection.

A. Outlier removal

When tracking an object, not every frame produces an accurate position of the tool tip. Since the detection method is rerun multiple times (i.e., every time when 75% of the interest points are lost), and the video is subject to visual artifacts, the detected tip position might be inconsistent throughout the video.

By assuming that the tool trajectory will be reasonably smooth, we can discard some of these inconsistencies through an outlier removal filter. This filter can be applied, individually, to the x and y coordinates of the tool position. In our implementation, we use a moving window w, and for each coordinate p_w within this window, we verify whether the following condition holds:

$$\frac{(p_w - \mu_w)}{std_w} < T_w$$

where μ_w and std_w represent the mean and the standard deviation of w, respectively, and T_w represents a threshold value. If this condition is true, p_w is considered valid. Otherwise, p_w is assigned the value μ_w. The result following the outlier filter being applied to the needle driver trajectory is illustrated in Figure 6. For our experiments, the threshold value T_w was based on the comparison with ground truth data. We basically defined a range of possible values, and measured the mean distance of the filtered data to the ground truth. We use the threshold value that produced the best results. The same value can be applied to other recordings, for which no ground truth data is available.

III. RESULTS

To validate our method, the accuracy of the tracked data was measured by using ground truth data, manually defined for each frame from one of the trials. The video used in our validation method has the resolution of 1280x720 pixels. The comparison of the ground truth with the tracking data was obtained through a MATLAB script. We measured both the distance of the ground truth to the tracked tool tip, as well as the correlation between their trajectories. Considering the trajectory, we achieved a 0.9978 correlation value, using the Pearson scale, which can be considered an extremely high correlation value, and it demonstrates the robustness of our tracking algorithm. The comparison between the ground truth and tracked tool position is illustrated in Figure 7. The mean euclidean distance between the two trajectories is 43 pixels. It can be observed that tracking is still lost for some short periods of time. This happens especially when there is an abrupt position change. Since the mean value is more sensitive to outliers in the distribution, we also computed the median value, which is 12 pixels for the driver.

IV. CONCLUSION

In this work we proposed an offline method to estimate the position of the tool tip in microsurgical training recordings. This method requires no special experimental setup, and it is robust to visual artifacts. It uses known features of a tool, detected in the video, and subsequently tracks the detected feature points, without relying on background subtraction algorithms, which are not feasible for non-static backgrounds. Furthermore, our method requires no training data, and it can easily be employed in other datasets. In future work, we intend to extend this method to capture additional motion information, such as tool rotation.

ACKNOWLEDGMENT

We thank NSERC, Canada, and Alberta Innovates Technology Futures (AITF) for their financial support.

REFERENCES

[1] M. Singh and A. Saxena, "Microsurgery: A useful and versatile tool in surgical field," *Surgery: Current Research*, pp. ISSN: 2161–1076, April.

[2] A. Ghousia, M. Prabhuji, and R. Lavanya, "Microsurgery: A clinical philosophy for surgical craftsmanship," *Ejournal of Dentistry*, no. 2, pp. 233–237, 2012.

[3] L. Richstone, M. J. Schwartz, C. Seideman, J. Cadeddu, S. Marshall, and L. R. Kavoussi, "Eye metrics as an objective assessment of surgical skill," *Annals of surgery*, vol. 252, no. 1, pp. 177–182, 2010.

[4] M. Wilson, J. McGrath, S. Vine, J. Brewer, D. Defriend, and R. Masters, "Psychomotor control in a virtual laparoscopic surgery training environment: gaze control parameters differentiate novices from experts," *Surgical endoscopy*, vol. 24, no. 10, pp. 2458–2464, 2010.

[5] N. Stylopoulos and K. G. Vosburgh, "Assessing technical skill in surgery and endoscopy: a set of metrics and an algorithm (c-pass) to assess skills in surgical and endoscopic procedures," *Surgical Innovation*, vol. 14, no. 2, pp. 113–121, 2007.

[6] G.-Q. Wei, K. Arbter, and G. Hirzinger, "Automatic tracking of laparoscopic instruments by color coding," in *CVRMed-MRCAS'97*. Springer, 1997, pp. 357–366.

[7] X. Jiang, B. Zheng, and M. S. Atkins, "Video processing to locate the tooltip position in surgical eye–hand coordination tasks," *Surgical innovation*, p. 1553350614541859, 2014.

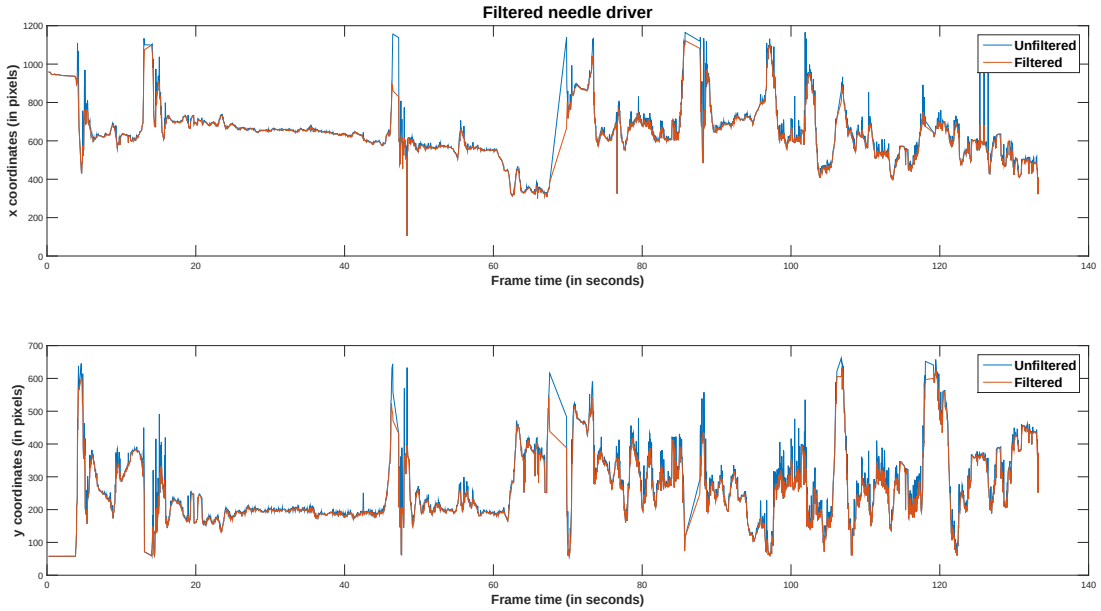

Fig. 6. Application of the outlier filter to the needle driver trajectory. For easier visualization, trajectory is segmented: x-axis (Top image) and y-axis (Bottom image).

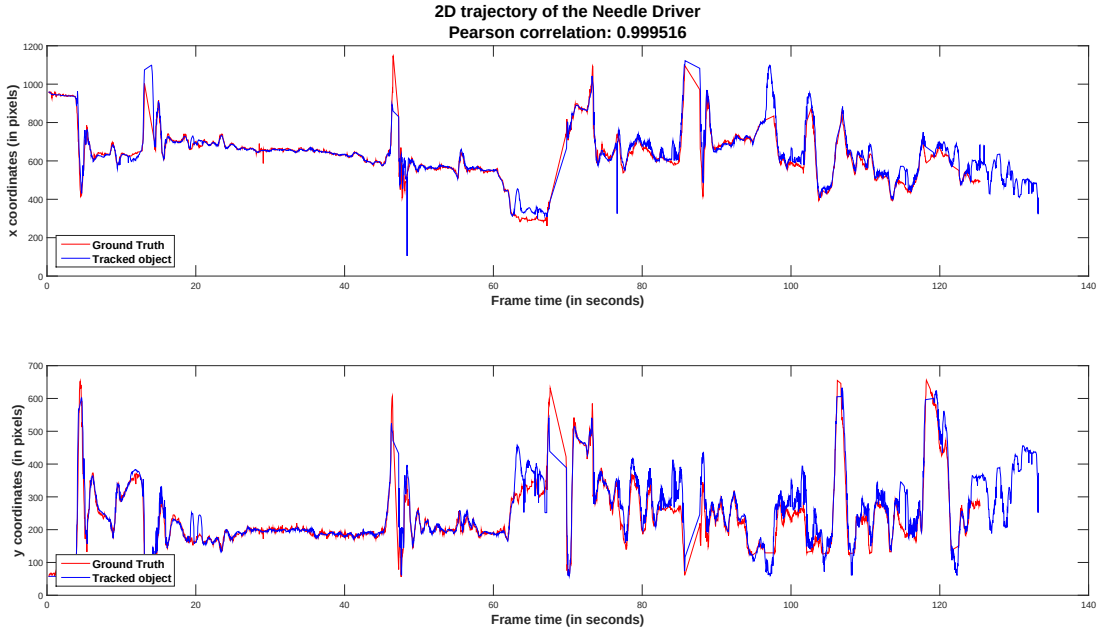

Fig. 7. Trajectory of the tracked needle driver is compared to ground truth data. For easier visualization, trajectory is segmented: x-axis (Top image) and y-axis (Bottom image).

[8] Z. Pezzementi, S. Voros, and G. D. Hager, "Articulated object tracking by rendering consistent appearance parts," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3940–3947.

[9] F. Pérez, H. Sossa, R. Martínez, D. Lorias, and A. Minor, "Video-based tracking of laparoscopic instruments using an orthogonal webcams system," *World Acad Sci Eng Technol Int J Med Health Pharm Biomed Eng*, vol. 7, no. 8, pp. 184–187, 2013.

[10] F. Meyer, "Topographic distance and watershed lines," *Signal processing*, vol. 38, no. 1, pp. 113–125, 1994.

[11] C. Harris and M. Stephens, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[12] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, vol. 81, 1981, pp. 674–679.

[13] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.