Feature Classification for Tracking Articulated Surgical Tools

Austin Reiter[1], Peter K. Allen[1], and Tao Zhao[2]

[1] Columbia University, New York NY USA
{areiter,allen}@cs.columbia.edu
[2] Intuitive Surgical, Inc, Sunnyvale CA USA
tao.zhao@intusurg.com

Abstract. Tool tracking is an accepted capability for computer-aided surgical intervention which has numerous applications, both in robotic and manual minimally-invasive procedures. In this paper, we describe a tracking system which learns visual feature descriptors as class-specific landmarks on an articulated tool. The features are localized in 3D using stereo vision and are fused with the robot kinematics to track all of the joints of the dexterous manipulator. Experiments are performed using previously-collected porcine data from a surgical robot.

Keywords: Tool Tracking, Surgical Robotics, Learning, Features, Fusion.

1 Introduction

Robotic surgery has become widely used in recent years as it has enhanced the inherent abilities of the human surgeon. For example, there are more than 1800 da Vinci® [1] surgical systems working in operating rooms all over the world that performed about 360,000 surgical procedures in 2011. High definition stereo vision helps the surgeon see the anatomy and interact with the surgical tools with great clarity. Augmenting the surgeon's vision with other relevant information in the form of graphical overlays can further help the surgeons/patients in a different dimension.

Knowledge of the locations of tools in the endoscopic video can enable a wide spectrum of applications. Virtual measurements can be utilized to provide 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 at the tip of the instrument which is close to the surgeon's visual center of attention, enhancing the safety of using such tools. It is also useful in managing the tools that are off the screen, increasing patient's safety, or for visual servoing of motorized cameras.

The joints of a robotic surgical system are usually equipped with encoders so that the pose of the end effectors can be computed using forward kinematics. On one hand, 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. On the other hand, a master-slave system 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

N. Ayache et al. (Eds.): MICCAI 2012, Part II, LNCS 7511, pp. 592–600, 2012.

mentioned above. Therefore, detecting and tracking the tools from images is a practical way to achieve the accuracy requirements of the applications.

Previous approaches to tool tracking have employed specialized fiducial markers to locate the tool [2,3,4]. Effective as they are, there are practical challenges such as manufacturability and cost. Alternatively, marker-less techniques have used: color segmentation to label pixels in the image as tool against the background [5,6,7]; geometric priors to confine the search space from the abdominal wall [8,9,10]; or combining different features together to detect the tool in the image [11,12,13]. Most prior work emphasizes the shaft, however in robotic surgery surgeons tend to work very close to the anatomy and the small visible part of the shaft may cause these algorithms to work poorly. This has motivated us to investigate the features on the tip of the tools.

In this paper we present a tracking system which learns the appearances of natural landmarks on an articulated tool by training an efficient multi-class classifier on a discriminative feature descriptor. We run the classifier on an image frame to detect all extrema representing the location of each feature type, where confidence values help reject false positives. We stereo match in the corresponding camera to recover 3D locations. By also knowing these landmark locations on the tool CAD model, we recover a pose offset of the kinematics using an Extended Kalman Filter. This fusion of vision and kinematics fills in the gap of missed vision detections and the articulations that are not estimated by vision, making the estimation of the tool pose available at all times.

2 Materials and Methods

2.1 Training Data Collection

We begin by collecting data to train our classifier. We use five different video sequences which span various in-vivo experiments, to best cover a range of appearance and lighting scenarios. We concentrate on the Large Needle Driver (LND) tool, keeping in mind this technique may be applied to any other types of tools. Seven visually salient and stable landmarks are manually selected and shown on the left of Fig. 1. The features chosen are of the pins that hold the distal clevis together, the **IS** logo in the center, and the wheel and wheel pin. We also know these locations on the tool's CAD model, which will be used for association to compute the final articulated pose.

For each frame in the labeling procedure, we manually drag a bounding-box around each feature of interest, being careful to avoid contamination from pixels which don't belong to the tool. Overall, we use $\sim 15,000$ training samples across the 7 classes.

2.2 Feature Descriptor

There has been a large amount of prior work on visual feature detection and matching in the computer vision community. Scale and affine invariant feature descriptors [15] have been very successful in matching planar features. However, we found that they work poorly for features on metal surfaces with lighting changes, as in the case of surgical tools with varying poses and light directions. The right-side of Fig. 1 shows example appearance changes typically encountered of the *IS_Logo* feature. We require

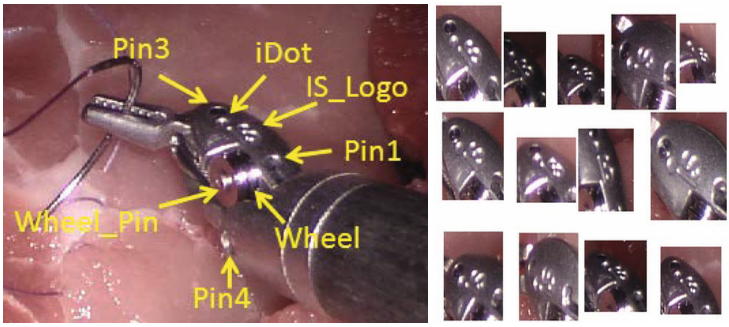


Fig. 1. [**Left**] The feature classes we detect on the LND tool. We concentrate on 7 different types of naturally-occurring landmarks. [**Right**] Examples appearance changes of the *IS_Logo* feature.

a discriminative and robust descriptor for our feature classes as each is fairly small (e.g., 17-25 pixels wide), and so we chose the Region Covariance Descriptor [16], where the covariance matrix of d features in a small image region serves as the feature descriptor. Given an image I, we extract d=*11* features, resulting in the feature image $\mathbf{F}$:

$$\mathbf{F} = [x \quad y \quad \mathbb{H} \quad \mathbb{S} \quad \mathbb{L} \quad I_x \quad I_y \quad I_{xx} \quad I_{yy} \quad \sqrt{I_x^2 + I_y^2} \quad \arctan(I_y/I_x)] \tag{1}$$

where x, y are the pixel locations; $\mathbb{H}, \mathbb{S}, \mathbb{L}$ are the hue, saturation, and luminance values at pixel location (x, y); I_x, I_y are the 1^{st}-order spatial derivatives; I_{xx}, I_{yy} are the 2^{nd}-order spatial derivatives; and the latter two features are the gradient magnitude and orientation, respectively. The covariance matrix $C_{\mathbf{R}} \in \mathbb{R}^{d \times d}$ of an arbitrary rectangular region $\mathbf{R}$ within F then becomes our feature descriptor.

Each $C_{\mathbf{R}}$ can be computed efficiently using integral images. We compute the sum of each feature dimension as well as the sum of the multiplication of every two feature dimensions. Given these first and second order integral image tensors, it can be shown that the covariance matrix of *any rectangular region* can be extracted in $O(d^2)$ time [16]. Using the ground truth data from Sec. 2.1, we extract covariance descriptors of each feature and store the associated feature label for training a classifier.

2.3 Feature Classification

There are several multi-class classifiers which may suit this problem. We adapt a method called *Randomized Trees* (RTs) [17] due to its computational efficiency. In addition to providing feature labels, we would like to retrieve confidence values for the classification task which we use to construct class-conditional likelihood images for each class.

RTs naturally handle multi-class problems very efficiently while retaining an easy training procedure. The RT classifier Λ is made up of a series of L randomly-generated trees $\Lambda = [\gamma_1, \ldots, \gamma_L]$, each of depth m. Each tree γ_i, for $i \in 1, \ldots, L$, is a fully-balanced binary tree made up of internal nodes, each of which contains a randomly-generated test that splits the space of data to be classified, and leaf nodes which contain estimates of the posterior distributions of the feature classes.

To train, the training features are dropped down the tree, performing binary tests at each internal node until a leaf node is reached. Each leaf node contains a histogram of length equal to the number of feature classes b. The histogram at each leaf counts the number of times a feature with each class label reaches that node. At the end of the training session, the histograms are normalized into probabilities using the total number of hits at that node. A feature is then classified by dropping it down the trained tree, again until a leaf node is reached, and is assigned the probabilities of belonging to a feature class depending on the posterior distribution stored at the leaf from training.

Because it's computationally infeasible to perform all possible tests of the feature, L and m should be chosen so as to cover the search space sufficiently and avoid randomness. Although this approach has been very successful for matching keypoints [17], traditionally the internal node tests are performed on a small patch of the gray image by randomly selecting 2 pixel locations and applying a binary operation ($\leq$) to determine which path to take to a child. In our problem, we are using feature descriptor vectors rather than image patches, and so we must adapt the node tests to suit our problem.

To this end, we use a similar approach to [18] for the node tests. For each internal tree node we construct a random linear classifier $h_i(\mathbf{x})$ on feature vector $\mathbf{x}$ to split:

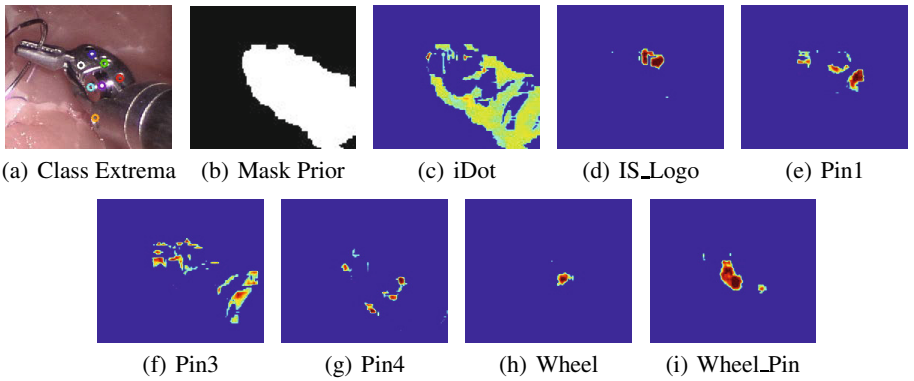
$$h_i(\mathbf{x}) = \begin{cases} \mathbf{n^T x} + z \leq 0 & \text{go to right child} \\ otherwise & \text{go to left child} \end{cases} \tag{2}$$

where $\mathbf{n}$ is a randomly generated vector of the same length as feature $\mathbf{x}$ with values in the range $[-1,1]$ and $z \in [-1,1]$ is also random. We found that this node test allows for robust splitting of the data and is computationally efficient. In this way, we build up probability distributions at the leaf nodes with the training descriptors. The results from each γ_i are averaged across the L trees. However, the d-dimensional nonsingular covariance matrices cannot be used *as is* to perform this task directly because they do not lie on a vector space, but rather on a connected Riemannian manifold. Due to space limitations, we refer the reader to [19] for a mathematical overview on post-processing the covariance descriptors to a Euclidean vector-space for use with a classifier. In the end, our $[d \times d]$ dimensional matrices $\mathbf{C_R}$ are mapped to vectors $\mathbf{c}_j \in \mathbb{R}^{d(d+1)/2}$.

2.4 Feature Class Labeling and Reconstruction

Labeling. Given our trained classifier Λ, we detect features for each label in an image by computing covariances $\mathbf{C_R}$, each of which is mapped to a vector space, producing $\mathbf{c}_j$. We drop each $\mathbf{c}_j$ through the trees γ_i and average the probabilities at the obtained leaf nodes to get a probability distribution p_L, representing the probability of $\mathbf{c}_j$ belonging to each of the L feature classes, resulting in L class likelihood images. To get the pixel locations, we perform non-maximal suppression in each class likelihood image. Example detections and likelihoods are shown in Figs. 2(a) and 2(c)-2(i), respectively.

Although the integral images afford efficient extractions of the covariances, we can reduce the computations further by initially segmenting the pixels of the image to identify areas of interest to classify. Using the method in [11], we train a Gaussian Mixture Model of several color and texture features to classify pixels into 1 of 3 groups (metal,



(a) Class Extrema (b) Mask Prior (c) iDot (d) IS_Logo (e) Pin1

(f) Pin3 (g) Pin4 (h) Wheel (i) Wheel_Pin

Fig. 2. Example likelihood images along with 6/7 successfully detected feature classes correctly located as extrema in (a). The *Pin3* (f) feature is incorrectly localized (white circle). The color-coding for the circles in (a) is: **Blue** for iDot, **Green** for IS_Logo (d), **Red** for Pin1 (e), **Orange** for Pin4 (g), **Purple** for Wheel (h), and **Cyan** for Wheel_Pin (i). A mask prior is shown in (b) which detects pixels on the metal part of the tool to reduce the number of pixels needing classification.

shaft, and background) and use the metal likelihood to produce a binary image of pixels to run through our classifier Λ. Fig. 2(b) shows a mask prior for the video image in 2(a).

3D Reconstruction. Now that we have candidate pixel locations for each feature class, we retrieve the 3D locations by stereo matching the feature tracks in the corresponding stereo camera using normalized cross-correlation checks along the epipolar line and triangulating the features. These feed into our fusion stage, described next.

2.5 Vision and Kinematics Fusion

The robot kinematics data can be fused with the vision to fill in the gap of the missed detections and to facilitate rejection of outliers. Many surgical robots like the da Vinci® need to maintain a stationary insertion point (also termed *remote center of motion*, or RCM). The errors in the passive setup joints accumulate at the RCM, resulting in a pose offset between the actual pose of the RCM and the pose computed using forward kinematics. This pose offset should be constant or slowly changing for a given surgical setup. Therefore, we can fuse vision and kinematics by solving this 6-DOF pose offset.

We denote **RCS** as the coordinate system of the true RCM and **KCS** as the coordinate system of the RCM according to the kinematics. The coordinates of a point $\mathbf{p}$ in both coordinate systems are associated by $\mathbf{p}^K = \mathbf{R}_R^K \mathbf{p}^R + \mathbf{c}_R^K$. We use an Extended Kalman Filter (EKF) to estimate the rigid transformation $\mathbf{R}_R^K, \mathbf{c}_R^K$ due to its adaptive nature, computational efficiency and the ability for uncertainty propagation. The information form of the filter is chosen to easily deal with a varying number of detected features. Also, the filter does not require the solution to be fully determined in a single frame. We omit the details of the implementation due to space limitations and refer the interested reader to [20] for more details.

It is possible that the output of the feature classification contains outliers. We employ RANSAC to enforce the rigid transformation using a sliding window approach and so

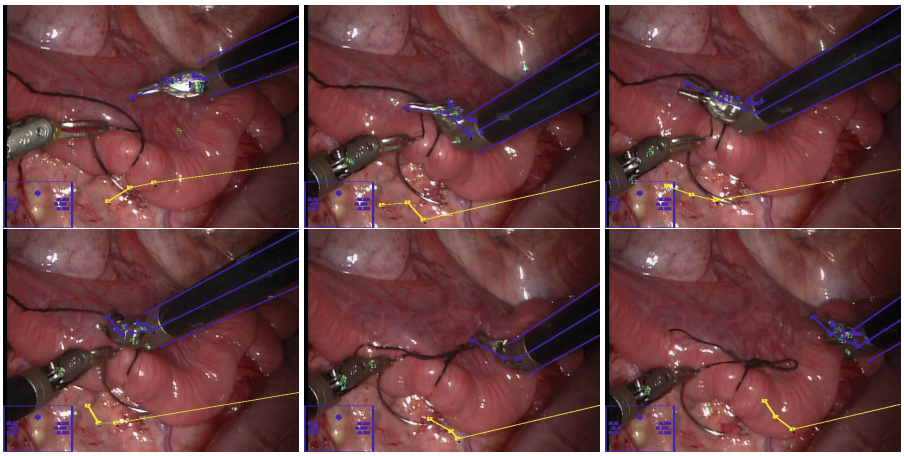


Fig. 3. Examples from our fusion tracker with the overlays for both the raw (yellow) and corrected (blue) kinematics. A visual inspection shows the tracker fixing the kinematics quite accurately.

only the inliers are fed into the EKF. We require a minimum of $\sim 30-50\%$ inliers to begin the filtering procedure.

3 Results

We experimented on previously collected porcine data from a da Vinci® surgical robot. The data which was used to test was specifically not included in the training procedure. After the off-line training of our RT classifier Λ (Sec: 2.3) using the seven feature classes shown in Fig. 1, we detected features (Sec. 2.4) and performed 3D reconstruction. These point locations were fed into the fusion module (Sec. 2.5) and the final kinematics joint overlay is drawn in the original image frame.

Six sample results are shown in Fig. 3, in which the yellow lines represent the raw kinematics projected into the image frames and the blue lines show the *fixed* kinematics resulting from our tracker. Notice the significant errors in some cases, motivating the need for the ideas presented in this paper. A visual inspection yields a fairly accurate correction of the kinematics overlaid on the right-most tool. The bottom-row middle-column shows a case where no features are visible, yet the EFK can predict because the remote center bias was previously accounted for correctly and remained static. This affords longer overall tracking times. On average it required $\sim 10-12$ frames for the fusion module to collect enough evidence to lock on to the true instrument kinematics.

Accuracy. The accuracy requirements for tool tracking depends on the application. Some applications, such as virtual measurement, may require $\leq$ 1mm accuracy while others such as status overlay can be more tolerant. It is regarded as acceptable if the estimated tool centerline is within the tool shaft, so it is not confused with another tool. This has motivated our evaluation method shown in Fig. 4, without being able to

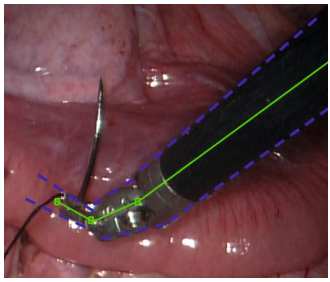


Fig. 4. To evaluate our kinematics estimates, the projected overlays must fall within the boundaries labeled as dotted blue lines here.

obtain ground truth in-vivo. The dotted blue lines define an acceptable boundary for the camera-projection of the kinematics. We manually inspect each frame of the test sequences, and resulted in a 93% accuracy rate over 1600 test frames. We attribute the incorrect estimates to the initial time necessary to lock on to the tool while the fusion module gathers evidence and occasional poor localization of the 2D features.

Occasionally we notice that the estimate is accurate on the tool tip, but slightly offset on the shaft. This occurs because we only include one feature on the shaft (*Pin4*), but in the future we will look to include more shaft information. Concurrent with this work we performed a study [21] on the feature detection accuracy, where we obtained an average localization accuracy of 86%, although this varies depending on the feature type. We also note that although some feature types are not always detected, we need only $\sim 3-4$ on a given frame because of the fusion, and so across the 7 chosen landmarks our experiments show that the percent correct achieved is sufficient for long-term tracking.

Timing. The tracker runs at ~ 1.2 secs/frame, where the feature detection takes most of the processing. This is dependent on: number of trees in Λ (90) , depth of each tree γ_i (10), number of features used in $\mathbf{C_R}$ (11), and the quality of the initial segmentation. It is also worth mentioning that we are estimating a pose offset that changes very slowly, therefore it is not critical that the estimation runs at frame-rate. Future speed-ups can come from GPU enhancements, because many parts of the algorithm are inherently parallel, and *tracking* of the features (between consecutive frames, the features shouldn't move far and so we only need to classify small windows around previous detections).

4 Conclusion

In this paper we have presented a method to learn to detect naturally-occurring landmarks on an articulated surgical tool to track it over time. We showed robustness in both feature detection and fusion of the vision observations with the inaccurate robot kinematics. Future work includes testing on different types of tools as well as tracking multiple tools in the scene simultaneously.

References

1. Intuitive Surgical, Inc., http://www.intuitivesurgical.com/
2. Wei, G.-Q., Arbter, K., Hirzinger, G.: Automatic Tracking of Laparoscopic Instruments by Color Coding. In: 1st Joint Conf. on Computer Vision, Virtual Reality, and Robotics in Medicine and Medical Robotics and Computer-Assisted Surgery, pp. 357–366 (1997)
3. Krupa, A., Gangloff, J., Doignon, C., de Mathelin, M., Morel, G., Leroy, J., Soler, L., Marescaux, J.: Autonomous 3-D Positioning of Surgical Instruments in Robotized Laparoscopic Surgery Using Visual Servoing. IEEE Trans. on Robotics and Automation 19(5), 842–853 (2003)
4. Groeger, M., Arbter, K., Hirzinger, G.: Motion Tracking for Minimally Invasive Robotic Surgery, pp. 117–148. Medical Robotics ITech Education and Publishing (2008)
5. Lee, C., Uecker, D.: Image Analysis for Automated Tracking in Robot-Assisted Endoscopic Surgery. In: Intl. Conf. on Computer Vision and Image Processing (1994)
6. Doignon, C., Nageotte, F., de Mathelin, M.: 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)
7. Doignon, C., Graebling, P., de Mathelin, M.: Real-time segmentation of surgical instruments inside the abdominal cavity using a joint hue saturation color feature. Real-Time Imaging (2005)
8. Doignon, C., Nageotte, F., de Mathelin, M.: Segmentation and Guidance of Multiple Rigid Objects for Intra-operative Endoscopic Vision. In: Vidal, R., Heyden, A., Ma, Y. (eds.) WDV 2005/2006. LNCS, vol. 4358, pp. 314–327. Springer, Heidelberg (2007)
9. Voros, S., Long, J.-A., Cinquin, P.: Automatic Detection of Instruments in Laparoscopic Images: A First Step Towards High-level Command of Robotic Endoscopic Holders. Intl. J. of Robotics Research 26(11-12), 1173–1190 (2007)
10. Wolf, R., Duchateau, J., Cinquin, P., Voros, S.: 3D Tracking of Laparoscopic Instruments Using Statistical and Geometric Modeling. In: Fichtinger, G., Martel, A., Peters, T. (eds.) MICCAI 2011, Part I. LNCS, vol. 6891, pp. 203–210. Springer, Heidelberg (2011)
11. Pezzementi, Z., Voros, S., Hager, G.: Articulated Object Tracking By Rendering Consistent Appearance Parts. In: IEEE Intl. Conf. on Robotics and Automation (2009)
12. Reiter, A., Allen, P.K.: An Online Approach To In-Vivo Tracking Using Synergistic Features. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (2010)
13. Sznitman, R., Basu, A., Richa, R., Handa, J., Gehlbach, P., Taylor, R.H., Jedynak, B., Hager, G.D.: Unified Detection and Tracking in Retinal Microsurgery. In: Fichtinger, G., Martel, A., Peters, T. (eds.) MICCAI 2011, Part I. LNCS, vol. 6891, pp. 1–8. Springer, Heidelberg (2011)
14. Burschka, D., Corso, J., Dewan, M., Lau, W., Li, M., Lin, H., Marayong, P., Ramey, N., Hager, G., Hoffman, B., Larkin, D., Hasser, C.: Navigating Inner-Space: 3-D Assistance For Minimally Invasive Surgery. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (2004)
15. Lowe, D.: Distinctive image features from scale-invariant keypoints. Intl. J. of Computer Vision 60(2), 91–110 (2004)
16. Tuzel, O., Porikli, F., Meer, P.: Region Covariance: A Fast Descriptor for Detection and Classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 589–600. Springer, Heidelberg (2006)
17. Lepetit, V., Fua, P.: Keypoint Recognition Using Randomized Trees. Trans. on Pattern Analysis and Machine Intelligence 28(9), 1465–1479 (2006)
18. Bosch, A., Zisserman, A., Muoz, X.: Image Classification using Random Forests and Ferns. In: IEEE Intl. Conf. on Computer Vision, pp. 1–8 (2007)

19. Tuzel, O., Porikli, F., Meer, P.: Human Detection via Classification on Riemannian Manifolds. In: IEEE. Conf. on Computer Vision and Pattern Recognition (2007)
20. Zhao, T., Zhao, W., Hoffman, B.D., Nowlin, W.C., Hui, H.: Efficient Vision and Kinematic Data Fusion for Robotic Surgical Instruments and Other Applications. US Patent US 20100331855 (June 2009)
21. Reiter, A., Allen, P.K., Zhao, T.: Learning Features on Robotic Surgical Tools. In: Workshop on Medical Computer Vision, CVPR 2012 (2012)